



BACHELOR THESIS

**ADVANCED ALGORITHMS FOR HIGH  
RESOLUTION TURBULENT STATISTICS  
WITH TOMOGRAPHIC PIV**

---

Author: Nereida Agüera  
Tutor: Stefano Discetti



# Contents

Abstract . . . . .	8
Resumen . . . . .	9
<b>1 Introduction</b>	<b>11</b>
1.1 Background . . . . .	11
1.2 Motivation . . . . .	15
1.3 State of the art . . . . .	19
1.4 Scope of the work . . . . .	20
<b>2 Methodology</b>	<b>23</b>
2.1 2D Ensemble PTV Algorithm . . . . .	23
2.1.1 Image Acquisition . . . . .	24
2.1.2 Particle Identification . . . . .	25
2.1.3 Particle Matching . . . . .	26
2.1.4 Ensemble Creation . . . . .	27
2.1.5 Ensemble Averaging . . . . .	27
2.1.6 Design alternatives: Use of filters . . . . .	28
2.2 3D Ensemble PTV Algorithm . . . . .	34
2.2.1 Image Acquisition . . . . .	34
2.2.2 Particle Identification . . . . .	35
2.2.3 Particle Matching . . . . .	36
2.2.4 Ensemble Creation . . . . .	36
2.2.5 Ensemble Averaging . . . . .	36
2.2.6 Design Alternatives: Use of Filters . . . . .	37
<b>3 Results and discussion</b>	<b>41</b>
3.1 2D Ensemble PTV Algorithm . . . . .	41
3.1.1 Validation: Boundary Layer Profile Test . . . . .	41
3.1.2 Performance Assessment of Mean Flow Field Estimation . . . . .	43
3.1.3 Performance Assessment of Turbulence Estimation. . . . .	47
3.2 3D Ensemble PTV Algorithm . . . . .	48
3.2.1 Validation . . . . .	48
3.2.2 Parametric Studies . . . . .	52
<b>4 Project Planning</b>	<b>55</b>
<b>5 Socioeconomic Context and Regulatory Framework</b>	<b>57</b>
5.1 Overview . . . . .	57
5.2 Project Cost . . . . .	58

<b>Conclusion</b>	<b>59</b>
<b>Bibliography</b>	<b>61</b>
<b>Acknowledgements</b>	<b>65</b>
<b>Appendix</b>	<b>66</b>

# List of Figures

1.1	Standard PIV experimental setup ( <a href="#">Raffel et al., 2007</a> ). . . . .	12
1.2	Schematic of optical off-axis setup in particle holography. <b>(a)</b> Recording of a hologram of a particle field; <b>(b)</b> Reconstruction of a virtual image; <b>(c)</b> Reconstruction of a real image ( <a href="#">Hinsch, 2002</a> )	13
1.3	Particle trajectory in image and object space ( <a href="#">Willneff and Gruen, 2002</a> ). . . . .	14
1.4	Working principle of Tomographic PIV ( <a href="#">Discetti (2013)</a> adapted from <a href="#">Elsinga et al. (2006)</a> ). . . . .	15
1.5	Spatial filtering of velocity profiles caused by window correlation ( <a href="#">Kähler et al., 2012</a> ). . . . .	16
1.6	Formation of ghost particles in a 2-camera setup ( <a href="#">Elsinga et al., 2011</a> ).	17
1.7	Possible reconstruction solutions to the 2-particle-2-camera problem of Figure 1.6 ( <a href="#">Elsinga et al., 2011</a> ). . . . .	17
1.8	Formation of ghost particles that contribute to the cross-correlation (left) and non-correlating ghost particle (right). Dark particles and solid lines-of-sight correspond to the first exposure. Bright particles and dashed lines-of-sight correspond to the second exposure ( <a href="#">Elsinga et al., 2011</a> ). . . . .	18
2.1	Sketch of the steps involved in 2D Ensemble PTV. . . . .	23
2.2	Sketch of the image generation process. . . . .	24
2.3	Example of an intensity map for a 2D image. . . . .	25
2.4	Sketch of the particle pairing process. . . . .	27
2.5	Ensemble creation: the velocity vectors from every image pair are all put together in a conglomerate. . . . .	27
2.6	Weighting distributions: <b>a)</b> Top-hat filter, <b>b)</b> Gaussian filter. . . . .	29
2.7	Schematic of the residual error formation: <b>a)</b> Top-hat filter, <b>b)</b> Gaussian filter, <b>c)</b> Polynomial filter. . . . .	32
2.8	Visual representation of the steps involved in the 3D Ensemble PTV method . . . . .	35
2.9	Sketch of the particle identification process. . . . .	35
3.1	Validation test results using a Boundary Layer profile flow field. . . . .	42
3.2	Total error variation with the number of images processed for a sinusoidal mean flow field. . . . .	44
3.3	MTF variation with the number of images processed for a sinusoidal mean flow field. . . . .	45
3.4	Variation of total error with normalised spatial frequency. . . . .	46

3.5	Variation of MTF with normalised spatial frequency. . . . .	46
3.6	Mean $u'^2$ profile. . . . .	47
3.7	Error maps: <b>a)</b> Top-hat filter, <b>b)</b> Gaussian filter, <b>c)</b> Polynomial filter. . . . .	48
3.8	Exact jet-like displacement flow field. . . . .	49
3.9	Results for the jet-like displacement flow field using 3D Ensemble PTV with a polynomial filter. . . . .	50
3.10	Mean flow field results for a synthetic test case involving a jet-like displacement on a slice perpendicular to the jet. . . . .	51
3.11	Comparison of results for the mean flow field using Tomo-PIV and Ensemble PTV with three different filters. . . . .	51
3.12	Comparison of exact distribution of turbulent fluctuations with the results using different filters for the jet-like displacement flow field. . . . .	52
3.13	Results for a synthetic test case involving a jet-like displacement flow field. Influence of the interrogation window size: <b>a)</b> Mean flow field, <b>b)</b> Reynolds stresses. . . . .	52
3.14	Results for a synthetic test case involving a jet-like displacement flow field. Influence of the number of images: <b>a)</b> Mean flow field, <b>b)</b> Reynolds stresses. . . . .	53

# List of tables

4.1	Gantt chart for project execution. . . . .	55
5.1	Labor cost. . . . .	58
5.2	Equipment cost. . . . .	58
5.3	Software cost. . . . .	59
5.4	Total cost. . . . .	59

## Abstract

Turbulence is a really common and easily observable phenomenon in nature yet it still represents one of the most important unsolved scientific challenges. At the present time, Tomographic Particle Image Velocimetry (Tomo-PIV) might have become the most versatile and accurate experimental method to investigate three-dimensional turbulent flows. However, the main weakness of the technique concerns spatial resolution due to the limited seeding density that can be achieved without drastically impacting the quality of the tomographic reconstruction and, consequently, of the measured velocity fields. In planar PIV, ensemble correlation methods allowed for a great increase in spatial resolution by averaging the correlation maps of multiple samples. An alternative approach, with potentially higher spatial resolution, consists on tracking particles on individual exposures and performing ensemble averages of the cloud of particle pairs on small regions. This method is typically referred to as Ensemble Particle Tracking Velocimetry (Ensemble PTV). In this work, an implementation of this technique, both for the two-dimensional and three-dimensional cases, is presented and validated by performing several synthetic tests. Besides, a possible optimization of the method through the use of filters is investigated. The results obtained demonstrate the enormous potential of Ensemble PTV, especially when compared to standard PIV in 2D and Tomographic PIV in 3D, suggesting that it might bring about a drastic increase in spatial resolution and therefore contribute to a better understanding of turbulence.



## Resumen

La turbulencia es un fenómeno común y fácilmente observable en la naturaleza, sin embargo, aún representa uno de los más importantes retos científicos sin resolver. Actualmente, la PIV Tomográfica podría haberse convertido en la técnica de medida experimental más versátil y precisa para estudiar flujos turbulentos. No obstante, la resolución espacial que se puede conseguir con esta técnica está limitada por la imposibilidad de aumentar la densidad de partículas sin perjudicar la calidad de la reconstrucción tomográfica y en consecuencia, la validez de los resultantes campos de velocidades. En PIV planar, se ha conseguido una mejora en la resolución espacial mediante el uso de nuevos métodos de correlación (*Ensemble correlation*) basados en calcular la media de los mapas de correlación obtenidos de múltiples pares de imágenes. Otro método alternativo, con un mayor potencial para aumentar la resolución espacial, consiste en localizar las partículas en cada una de las imágenes y realizar la media conjunta de todas ellas en pequeñas regiones. Este método es comúnmente denominado *Ensemble PTV*. En este proyecto, dicha técnica es implementada, tanto en 2D como en 3D, y también validada mediante la realización de varios tests. Además, se investiga una posible optimización basada en el uso de filtros. Los resultados obtenidos demuestran el enorme potencial de Ensemble PTV, especialmente notable al comparar con las medidas obtenidas con PIV estándar en 2D y Tomo-PIV en 3D, sugiriendo que este método podría suponer un aumento drástico de la resolución espacial y por tanto contribuir a una mejor comprensión de la turbulencia.



# Chapter 1

## Introduction

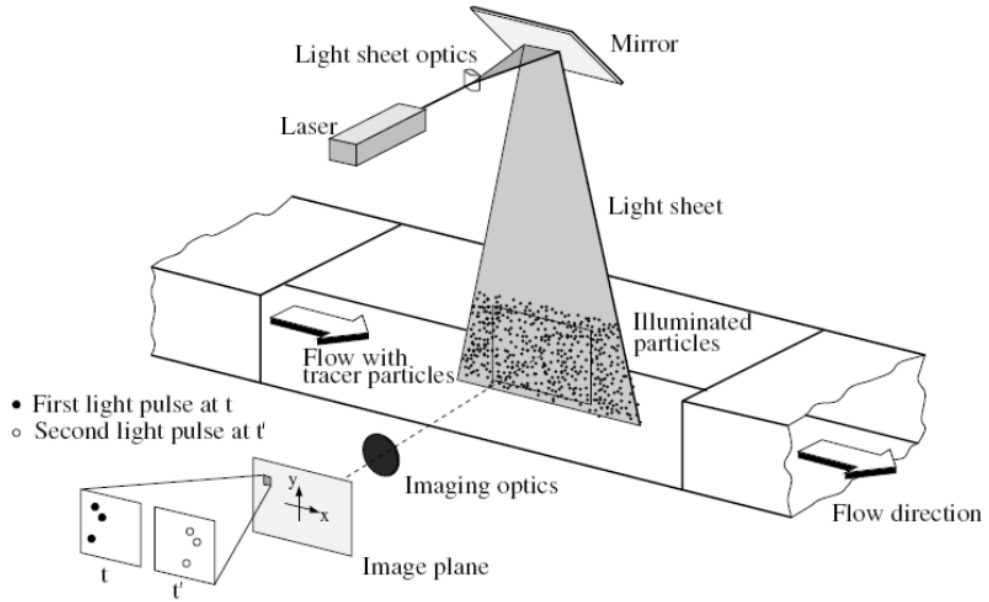
### 1.1 Background

Turbulence is a prevalent phenomenon in nature and a widely discussed topic in Fluid Mechanics. The vast majority of flows is turbulent and many engineering solutions have been made possible thanks to the inherent characteristics of turbulence, such as its capability to enhance the rate of mixing of matter, momentum and heat. Nonetheless, in spite of the numerous efforts devoted to it, the complete understanding of turbulence still remains far from reach ([Pope, 2000](#)).

In this quest for understanding, the advancement of technology has provided us with increasing computational capabilities that have triggered the development of the Computational Fluid Dynamics (CFD), enabling the use of Direct Numerical Simulation (DNS), among others. This technique relies on the well-known fact that Navier Stokes equations can fully describe both laminar and turbulent flows and therefore it uses them to numerically resolve all the scales of motion. However, it is worth mentioning that although this method offers an unprecedented level of accuracy and detail, it also involves an enormous computational cost. This is the reason why in spite of the fact that it constitutes a very valuable tool to model turbulent flows at moderate Reynolds numbers, it is practically inapplicable for high Reynolds numbers ([Pope, 2000](#)). The application of numerical methods on this high Reynolds number range needs to rely on turbulence models to solve the closure problem, like the Reynolds Average Navier-Stokes (RANS) equations or Large Eddy Simulations (LES), and that is why these models need to be validated by high resolution - high accuracy experimental benchmarks.

As far as experimental methods are concerned, major achievements have been made since Particle Image Velocimetry (PIV) was first conceived ([Adrian, 1984](#)). PIV is a non-intrusive measurement technique intended to estimate the velocity field of a fluid flow ([Raffel et al., 2007](#)). In order to do that, the fluid of interest is seeded with tracer particles and illuminated with a laser. Light is scattered by the particles and this is recorded by a camera at two or more successive instants of time. The information contained on each image pair is typically analyzed by dividing the frames into interrogation windows and using spatial correlation techniques. In this way, an average particle displacement is assigned to each interrogation

spot. Finally, since the time interval between the two frames is known, the velocity field distribution can be easily obtained. The process is depicted in Figure 1.1.



**Figure 1.1:** Standard PIV experimental setup ([Raffel et al., 2007](#)).

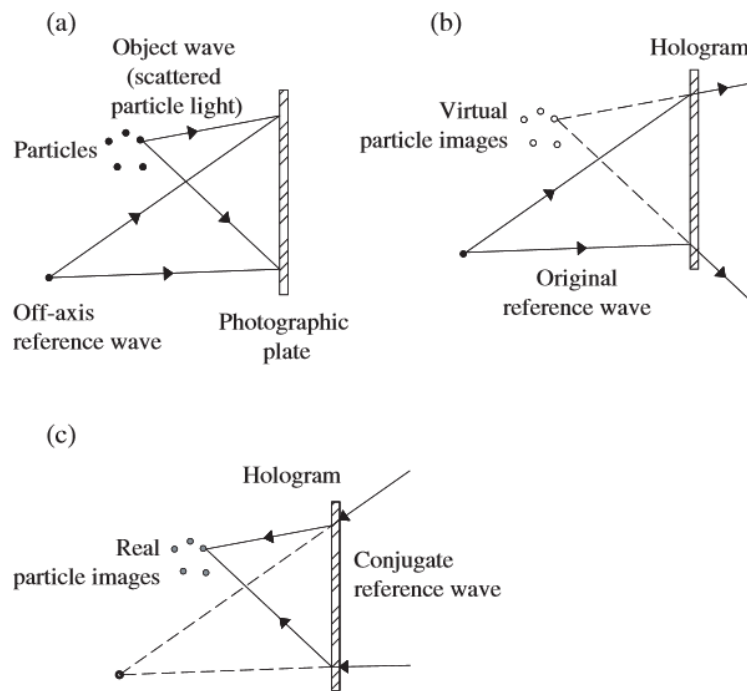
In the last decades, numerous PIV-based methods have been proposed to study both two-dimensional and three-dimensional flows. However, two-dimensional PIV only provides information about four of the components of the velocity gradient tensor, therefore needing to rely on assumptions to describe the three-dimensional behavior.

As a result, the development of extensions of well-established particle imaging methods towards higher dimensions has become the authentic challenge since that would allow us to measure the instantaneous three-dimensional (3D), three-component (3C) velocity field, therefore obtaining a full characterization of three-dimensional flows including unsteady coherent flow structures and turbulence. Among the many techniques developed with this goal in mind, the most prominent ones have been Scanning Light Sheet (SLS), Holographic PIV, Three-Dimensional Particle Tracking Velocimetry (3D PTV), and Tomographic PIV.

Scanning PIV ([Brücker, 1995](#)) combines classical PIV with volume scanning, using a high pulse frequency laser sheet to scan the volume of interest. It has potentially the same resolution as standard PIV and it is able to successfully describe transitional and turbulent flows ([Brücker, 1997](#), [Burgmann et al., 2006](#)). However, it is only applicable to moderate speed flows, since the time needed to sweep the volume must be smaller than the characteristic time scale of the flow. Therefore, high speed flows would require unachievable scanning frequencies.

Holographic PIV ([Collier et al., 1971](#)) involves the recording of the laser light-field scattered by the particles combined with a reference wave from the same laser

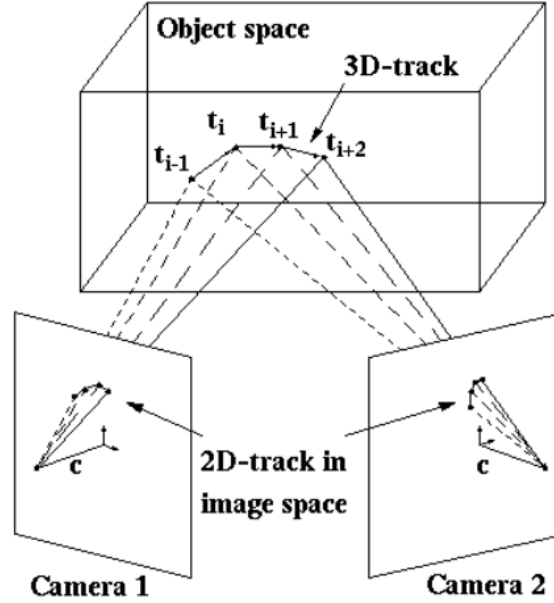
source. Then, the original wave field is reconstructed by illuminating the hologram (the processed interference pattern) with a conjugate reference wave. The process is depicted in Figure 1.2 and a comprehensive review is given by [Hinsch \(2002\)](#). This lens-less technique potentially allows for infinite depth of field; however, the recording medium imposes limitations in the amount of data that can be collected and makes it unsuited to study statistical flow properties. In order to overcome this issue, Digital Holographic PIV was developed ([Schnars and Jüptner, 1994](#), [Meng et al., 2004](#)). In this way, the hologram is directly recorded by a CCD-camera and then numerically reconstructed. Nonetheless, this method is still limited by the resolution of CCD cameras.



**Figure 1.2:** Schematic of optical off-axis setup in particle holography. **(a)** Recording of a hologram of a particle field; **(b)** Reconstruction of a virtual image; **(c)** Reconstruction of a real image ([Hinsch, 2002](#))

Three-dimensional Particle Tracking Velocimetry (3-D PTV) identifies and tracks particles' motion within subsequent frames. It has been referred to as a low density range of PIV ([Adrian, 1991](#)). However, unlike PIV, it is able to follow the particles in time in a Lagrangian sense, yielding instantaneous three-dimensional velocity vectors within the volume of interest and the particles' trajectories over long times ([Maas et al., 1993](#), [Malik et al., 1993](#)), provided high speed equipment is available and the flow is relatively slow. In order to do that, sets of 3D coordinates of particles are extracted from the analysis of multi-camera images of the flow region studied, as depicted in Figure 1.3.

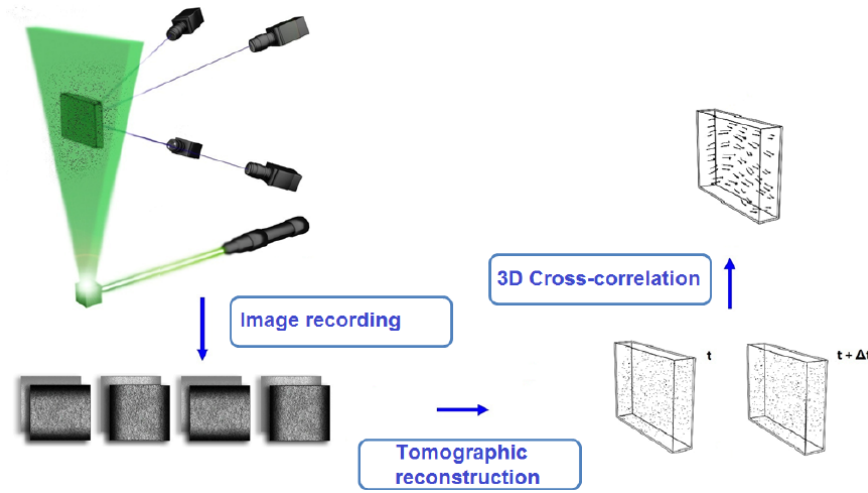
Tomographic PIV was implemented for the first time almost ten years ago ([Elsinga et al., 2006](#)) and since then, several research groups have been working on its optimization. Tomographic PIV allows to measure the full velocity vector



**Figure 1.3:** Particle trajectory in image and object space (Willneff and Gruen, 2002).

field in a 3D volume by means of a three-dimensional particle pattern correlation, thus extracting a spatially complete description of the instantaneous flow features and of the statistical characterization of turbulent flows. As sketched in Figure 1.4, multiple cameras image the volume of interest from different viewing angles and the two dimensional frames obtained are used to reconstruct a three dimensional light intensity field which is then subdivided in voxels, which are the three-dimensional counterpart of pixels. Then, a three-dimensional cross-correlation is performed between subsequent recordings on each voxel, therefore obtaining a full description of the velocity field (Elsinga et al., 2006).

Comparative studies between most of the methods presented above can be found in Arroyo and Hinsch (2008). Besides, Schaefer et al. (2011) presents a direct comparison between Holographic and Tomographic PIV. At the present time, it seems that Tomo-PIV could be the most promising one. The main advantage is that it can operate at higher seeding densities, typically 0.05 particles per pixel (Elsinga et al., 2006), which allows for a higher spatial resolution of the measurement. Indeed, it might have become the most versatile and accurate method to investigate three-dimensional flows to date.



**Figure 1.4:** Working principle of Tomographic PIV (Discetti (2013) adapted from Elsinga et al. (2006)).

## 1.2 Motivation

As explained in the previous section, Tomo-PIV has probably become the most robust and accurate method to describe three-dimensional flows. However, its main weakness resides in its limited spatial resolution.

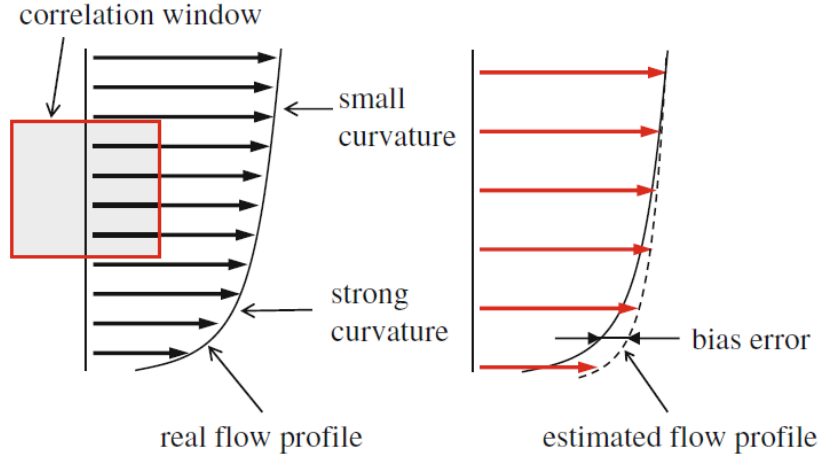
First of all, since Tomo-PIV is a correlation-based method, its spatial resolution is bounded by the size of the interrogation window, which should ideally be small relative to the characteristic length scales of the flow field. At each interrogation window, the particle displacement is determined from the location of the signal peak in the correlation plane. Then, the velocity is estimated using the fundamental definition of velocity shown in Equation 1.1 where  $\Delta S$  is the displacement and  $\Delta t$  is the time interval between the two illuminations. Some other factors, such as the magnification of the imaging system must be taken into account too.

$$V = \lim_{\Delta t \rightarrow 0} \frac{\Delta S}{\Delta t} \quad (1.1)$$

This implies that each measured velocity vector is not actually the flow velocity at that precise point but an average within the interrogation window. This is mathematically expressed in Equation 1.2 (Kähler et al., 2012), where  $\mathbf{u}$  is the measured velocity vector at  $\mathbf{r}$  and  $\mathbf{r}'$  is the location of the particle images. Here the weighting function  $G$  is affected by factors such as the illumination and imaging system, including the camera pixel size  $S$  or the laser beam profile, among others. On the other hand, the volume  $\Delta V$  is determined by the interrogation window size, the light sheet thickness and the particle size.

$$\langle \mathbf{u}(\mathbf{r}, t) \rangle = \int_{\Delta V} G(\mathbf{r}, \mathbf{r}', S) \mathbf{u}(\mathbf{r}', t) dV' \quad (1.2)$$

As a result of this averaging within the interrogation windows, significant bias errors appear, particularly when large flow gradients are present in the flow field, as depicted in Figure 1.5.



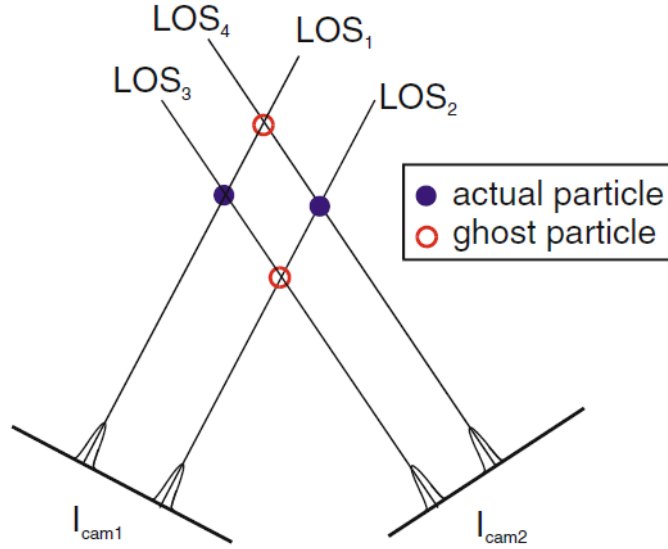
**Figure 1.5:** Spatial filtering of velocity profiles caused by window correlation (Kähler et al., 2012).

Following this line of thought, in order to increase the spatial resolution, one would think that reducing the size of the interrogation window would suffice, since that would lead to the obtention of independent velocity vectors as spatially close as needed. However, that assertion is incorrect, since there is a minimum number of particles that need to be contained in the interrogation window for the correlation to be successful. As a rule of thumb, the interrogation window should contain 7-10 particles. Then, for a given seeding density, there is a limit on the minimum window size that can be used and as a consequence, on the spatial resolution achievable. Therefore, the next logic step would be to increase the particle density. Nonetheless, for the results of Tomo-PIV to be successful, the particle image density should not exceed certain value, which is generally agreed to be 0.05 ppp (particles per pixel). Higher seeding densities lead to the proliferation of ghost particles, which appear during the tomographic reconstruction. A comprehensive study of this phenomenon can be found in Elsinga et al. (2011). Basically, ghost particles derive from the fact that Tomographic PIV is an undetermined problem, since a limited number of cameras is available and a three-dimensional distribution of particles must be reconstructed just by using the projection of the particles on the camera planes.

Figure 1.6 illustrates the phenomenon of ghost particles in a simple way by subtracting one dimension from the problem, in other words, it shows the problem of reconstructing a 2D distribution of particles from 1D projections. However, this simplified problem is also undetermined and it can be used to help understand the

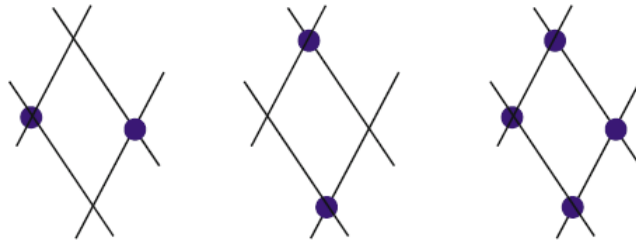


real phenomenon of ghost particles, in which a 3D distribution is reconstructed from 2D projections. More specifically, suppose that two particles need to be reconstructed using only two views taken from two different cameras. Camera 1 'sees' those two particles along the lines of sight  $LOS_1$  and  $LOS_2$ , while Camera 2 'sees' them through the lines of sight  $LOS_3$  and  $LOS_4$ .



**Figure 1.6:** Formation of ghost particles in a 2-camera setup (Elsinga et al., 2011).

The images taken by these cameras are used to perform the tomographic reconstruction of the 2D distribution and with the present configuration, this might lead to three different possible solutions, depicted in Figure 1.7.



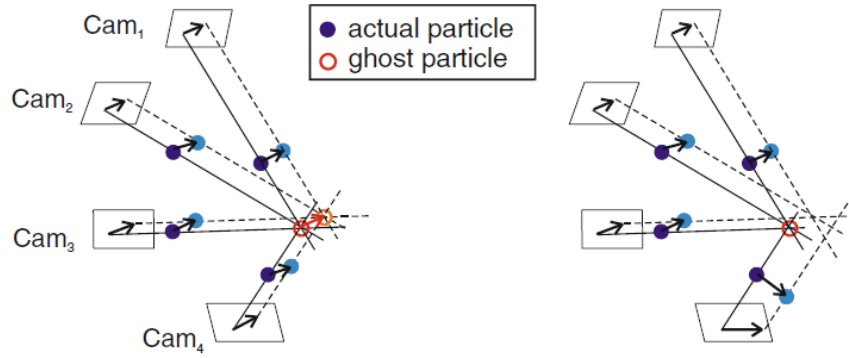
**Figure 1.7:** Possible reconstruction solutions to the 2-particle-2-camera problem of Figure 1.6 (Elsinga et al., 2011).

An estimate for the number of ghost particles can be found in (Discetti and Astarita, 2014) and it is obtained by taking into account that a ghost particle appears whenever within a search area as large as the particle image one can find a particle on all the camera images on the set and at least one of the projections is the image of another particle. This estimate corresponds to Equation 1.3, where  $N_g/N_p$  is the ratio of ghost particles to true particles,  $\epsilon_r$  is the particle radius,  $L_z$  is the volume depth in voxels -where a voxel is the three-dimensional equivalent of

the pixel-,  $N_{cam}$  is the number of cameras in the setup and  $N_s$  is the source density, defined as  $N_s = N_{ppp} \pi \epsilon_r^2$ .

$$\frac{N_g}{N_p} = N_{ppp} (2\epsilon_r L_z - \pi \epsilon_r^2) (1 - e^{-N_s})^{N_{cam}-2} \quad (1.3)$$

For instance, for  $N_{ppp} = 0.05 \text{ ppp}$ ,  $L_z = 200 \text{ vox}$ ,  $\epsilon_r = 1.5 \text{ px}$  and  $N_{cam} = 4$ , Equation 1.3 offers an estimate of  $N_g/N_p = 2.63$ , so that the number of ghost particles is more than two times larger than the number of true particles. Clearly, the number of ghost particles decreases with decreasing  $N_{ppp}$ . This is the main reason why the spatial resolution of Tomo-PIV cannot be enhanced by using a higher seeding density, since this would increase the number of spurious matches and decrease the accuracy of the method. Finally, it can also be seen that the number of ghost particles increases with the depth of the reconstructed volume.



**Figure 1.8:** Formation of ghost particles that contribute to the cross-correlation (left) and non-correlating ghost particle (right). Dark particles and solid lines-of-sight correspond to the first exposure. Bright particles and dashed lines-of-sight correspond to the second exposure (Elsinga et al., 2011).

The previous analysis deals with the occurrence of a ghost particle in a single volume; however, in order to affect the cross-correlation map and hence the velocity measurements, the ghost particle has to appear in both reconstructed volumes used in the cross-correlation analysis. As depicted in Figure 1.8, this occurs when the motion of the actual particles (the ones whose projection create the ghost particle) in the normal direction to the line of sight is nearly the same. If one of the particles presents a different displacement, the ghost particle no longer affects the correlation map.

Since the displacement of the spurious matches is approximately the average displacement of the associated actual particles, the effect of the ghost particles is to smooth and reduce the particle displacement variations and gradients over the volume thickness, since they spread the velocity information to points where there are no real particles. This does not alter the flow topology much but creates a bias

error due to the systematic underestimation of the velocity gradients.

To conclude, this section has briefly shown and explained the main weakness of Tomo-PIV, which corresponds to the limits on the achievable spatial resolution. It is evident that there exists an urgent need for a major breakthrough to drastically resolve this concerning issue of three-dimensional PIV.

### 1.3 State of the art

Due to the undeniable potential of Tomographic PIV, both as a method to investigate turbulence by itself and also as a benchmark for CFD results and validation, overcoming its spatial resolution limitations has become a topic of interest and concern, leading to the advent of new techniques.

Some solutions can be borrowed from 2D and then extended to the three-dimensional scenario. For instance, in planar PIV, ensemble correlation methods have allowed for a major increase in spatial resolution by averaging the correlation maps of multiple samples. With these techniques, the instantaneous information is lost, but temporal information is transferred into the spatial domain, thus allowing smaller interrogation regions that could potentially be as small as a single pixel, provided that the number of image pairs is large enough. That is why this method is generally referred to as 'Single Pixel Resolution Ensemble Correlation' (Westerweel et al., 2004). This method was applied for the first time by Westerweel et al. (2004) for stationary laminar flows in microfluidics. Then, it was applied to macroscopic flows, including laminar, turbulent and transitional (Kähler et al., 2006), compressible flows at large Mach numbers (Kähler and Scholz, 2006, Bitter et al., 2011) and after some more development, it was used to estimate Reynolds stresses in turbulent flows (Kähler et al., 2006, Scharnowski et al., 2012).

Single-Pixel Resolution Ensemble Correlation drastically reduces the spatial filtering and increases the spatial resolution, but it is not an exact solution. Although the size of the interrogation window can be as small as a pixel, it has been shown that the factor determining the spatial resolution is the particle-image diameter (Kähler et al., 2012). More specifically, the resolution is proportional to the typical length scale over which the particle image intensity changes, which is about half of the particle image diameter for Gaussian particle images. Besides, the particle image diameter  $d_\tau$  has a lower limit that is a function of the particle size  $d_p$ , the magnification of the imaging system  $M$ , the f-number  $f_\#$ , which is the ratio of the objective lens diameter and the aperture's diameter; the wavelength of the scattered light  $\lambda$ , the object's distance from the focal plane  $z$ , the lens aperture diameter  $D_a$  and the object distance  $s_0$ , as given by Equation 1.4 (Olsen and Adrian, 2000).

$$d_\tau = \sqrt{(Md_p)^2 + (2.44f_\#(M+1)\lambda)^2 + \left(\frac{MzD_a}{s_0+z}\right)^2} \quad (1.4)$$

The three components inside the square root of Equation 1.4 correspond to the geometric, diffraction and defocusing components of the enlargement of the particle image with respect to the physical particle. The latter can be neglected for macro-PIV, since  $z$  is usually 1-3 orders of magnitude smaller than  $s_0$  for well-aligned optical systems. Equation 1.4 shows that an increase of the magnification will eventually lead to an enlargement of the image particle diameter and therefore a reduction of the spatial resolution. Following this line of thought, [Kähler et al. \(2012\)](#) proves that the achievable spatial resolution with Single-Pixel Resolution Ensemble Correlation is bounded and sets this resolution limit to 1.84 px instead of a single pixel.

An alternative approach, with potentially higher spatial resolution, consists on tracking particles on individual exposures and performing ensemble averages of the cloud of particle pairs on small regions. This method is typically referred as Ensemble Particle Tracking Velocimetry. For PTV, the spatial resolution is no longer limited by the particle image diameter but by the error in the determination of the mean position of the particle images corresponding to a particle image pair ([Kähler et al., 2012](#)). Therefore, using higher magnification would increase the resolution of the measurement in spite of the increase in the particle image diameter and this means there is no theoretical limit on the distance between independent mean vectors. Finally, although it is true that this method gives up the instantaneous information, it has the huge potential of providing high resolution turbulent statistics for turbulence modelling. Besides, the results presented by [Kähler et al. \(2012\)](#) clearly demonstrate the still unexplored potential of this method.

## 1.4 Scope of the work

This project acknowledges the importance of obtaining a full understanding of turbulence and the issues related to the spatial resolution limits of the state-of-the-art flow measurement methods. Ensemble PTV, a measurement technique that might have the potential to overcome the current resolution limits, is explored, validated and tested for both the two dimensional and three dimensional scenarios. While the 2D case has already been assessed ([Kähler et al., 2012](#)), its 3D counterpart has not been explored so far, especially from the viewpoint of turbulence statistics. The core of this document is constituted by five chapters, which are briefly introduced below.

- Chapter 1 is composed of four sections. The first serves as a general introduction to flow measurement techniques, particularly focusing on PIV. The motivation section presents the problem: in spite of the huge potential of Tomo-PIV, its limited spatial resolution poses a serious concern. The state-of-the-art section compiles the work done by other authors to assess this problem and the present section summarizes the main points and structure of this work as a whole.

- Chapter 2 presents the method proposed, Ensemble PTV, explaining in detail all the steps of the implementation process. Besides, a possible enhancement of the algorithm through the use of filters is examined, both in 2D and in 3D.
- Chapter 3 shows the results of the synthetic tests performed. In 2D, three cases were considered: a boundary layer profile flow field, a sinusoidal mean flow field with no turbulence and a sinusoidal mean flow field with sinusoidal turbulent fluctuations. In 3D, a jet-like displacement flow field with a pseudo-shear layer with isotropic turbulence is simulated.
- Chapter 4 presents the project planning and the distribution of the work in time.
- Chapter 5 provides some insight into the current socioeconomic situation and regulatory framework, and it also includes an estimate of the project cost.

Finally, some conclusions are drawn and a brief summary of the main ideas discussed in this work is presented.



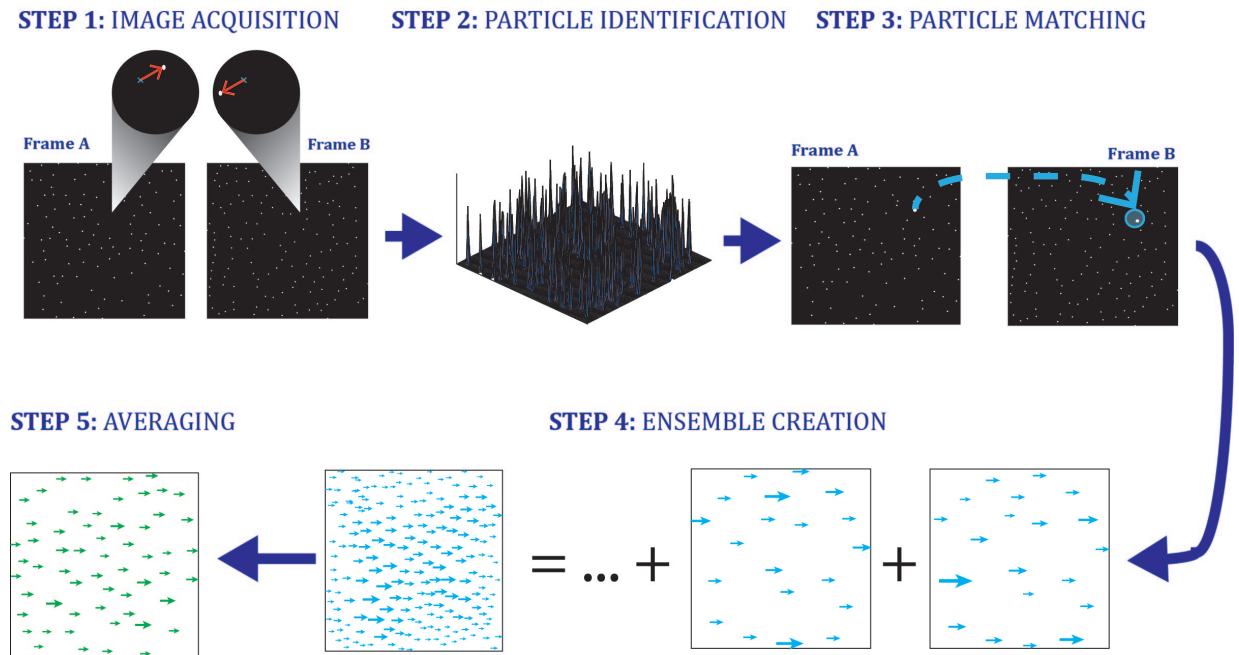
# Chapter 2

## Methodology

### 2.1 2D Ensemble PTV Algorithm

As previously discussed, 2D Ensemble PTV has already been explored by some authors ([Kähler et al., 2012](#)). In this chapter, a complete implementation of this concept is presented and explained in depth together with some potential ways of enhancing the results obtained by using weighting windows or polynomial fitting.

The whole process of 2D Ensemble PTV is summarized in Figure 2.1 for visual reference and can be divided into five steps.



**Figure 2.1:** Sketch of the steps involved in 2D Ensemble PTV.

### 2.1.1 Image Acquisition

The first step corresponds to the data acquisition. In this work, all test cases were synthetic, so this step actually corresponds to Image Generation instead of Image Acquisition. The images were virtually generated with the MATLAB code *GenImg.m* attached in the Appendix. This function creates images with the desired dimensions, number of particles and particle intensity. The particles are located randomly within the image dimensions and the desired flow field is described in a separate function, *VelField.m*, that is called by *GenImg.m*. Each particle, together with its random position is processed by *VelField.m* in a way that a replica of that particle is located on the two images or exposures that constitute an image pair. This is done in a centered fashion, as illustrated in Figure 2.2. In this way, on each exposure the particle is displaced half of the total displacement from the original random position, in such a way that the particle's displacement between the two exposures is the desired total displacement specified.

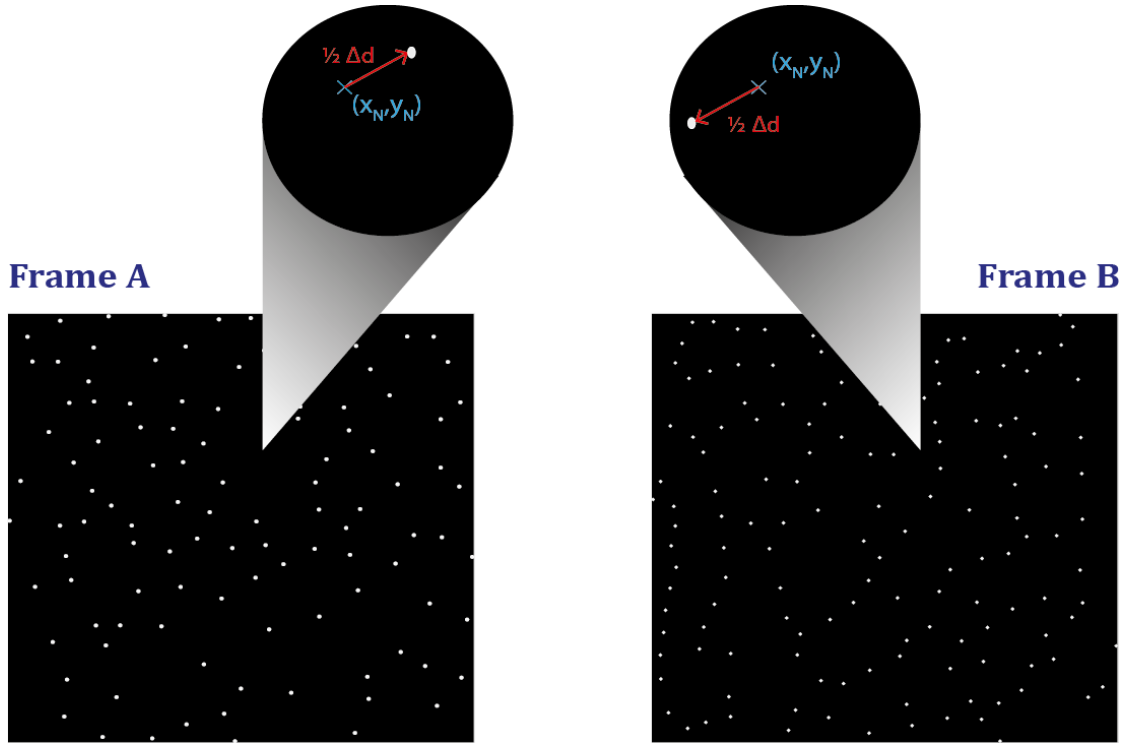


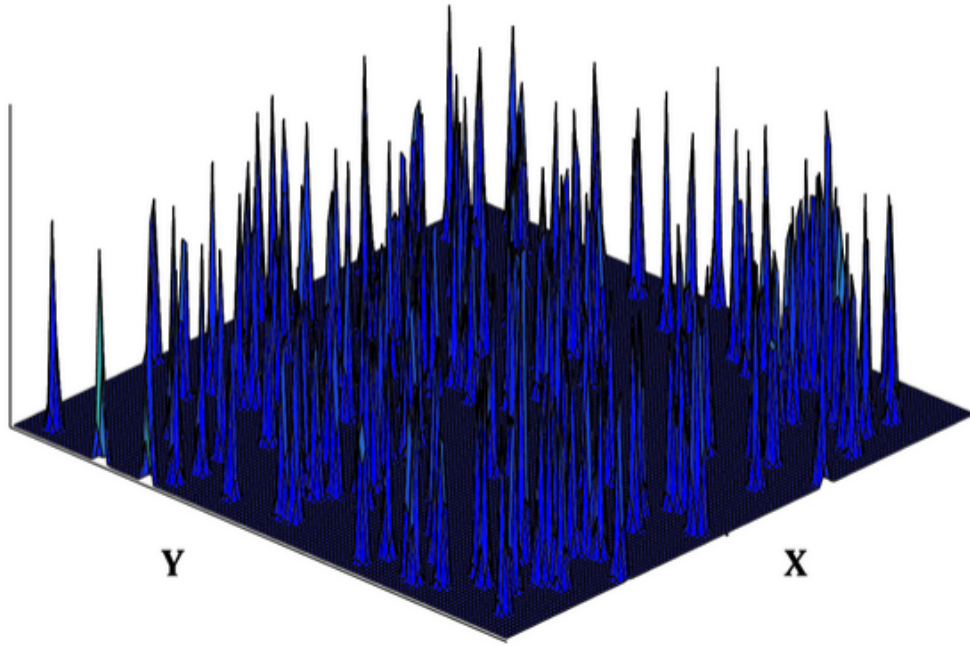
Figure 2.2: Sketch of the image generation process.

In all the experiments shown in this work, the distribution of intensity of the particles was Gaussian. However, one cannot directly place a Gaussian intensity distribution around the particle location in the synthetic image. In order to obtain a more realistic model of the imaging process of a CCD camera, it is necessary to account for the fact that due to the finite pixel size of the camera sensor, what truly appears in the camera images is not the continuous Gaussian intensity distribution but its integral on each pixel. In our synthetic images, this process is simulated by integrating the intensity discretely assuming an idealized camera sensor. This process is repeated for as many image pairs as needed for the experiment.



## 2.1.2 Particle Identification

Once the images have been generated, the first task is to locate the particles on each of the images taken. That is done by analyzing the intensity map on each image -an example of an intensity map is depicted in Figure 2.3-. Knowing that the particles will scatter the light and will correspond to the higher intensity locations, the local maxima locations on the intensity maps are found and then, Gaussian interpolation is used to obtain sub-pixel precision on the intensity peaks' locations and therefore, on the position of the particles.



**Figure 2.3:** Example of an intensity map for a 2D image.

The Gaussian interpolation is done by fitting a Gaussian function to the intensity matrix and deriving the maximum position of the interpolated function. In order to do this,  $3 \times 3$  kernels centered on each of the local maxima are isolated from the image intensity matrix. Then, it is assumed that the peak of the local maxima fits a 2D Gaussian function and that the two dimensions are separable and orthogonal. Then, since the logarithm of a Gaussian yields a second-order polynomial without changing the location of the maximum, the sub-pixel peak location is calculated separately for the two directions by fitting a second-order polynomial to the logarithm of the maximum sample and the two adjacent neighbours.

Equations 2.1 to 2.5 describe the process mathematically. Equation 2.1 represents the fitting function for the X-direction where  $a$ ,  $b$  and  $c$  are the unknown coefficients.  $I_{i,j}$  and  $x_{i,j}$  correspond to the logarithm of the intensity and x-location of pixel  $(i, j)$ , respectively. Equation 2.2 represents the location of the maximum intensity obtained from Equation 2.1. Then, expressing the logarithm of the intensity in terms of the fitting function, Equation 2.3 is obtained and it can

be solved for the polynomial coefficients. At this point, using Equation 2.2, the subpixel location in the X-direction,  $\Delta X$ , can be obtained, as shown in Equation 2.4. The same procedure can be followed for the Y-direction, arriving at Equation 2.5. This method allows for an increase of the precision on the peak location down to  $1/20$  of a pixel.

$$I_{i,j} = ax_{i,j}^2 + bx_{i,j} + c \quad (2.1)$$

$$x_{I,max} = \frac{-b}{2a} \quad (2.2)$$

$$\begin{bmatrix} I_{i-1,j} \\ I_{i,j} \\ I_{i+1,j} \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} a \\ b \\ c \end{bmatrix} \quad (2.3)$$

$$\Delta X = \frac{I_{i-1,j} - I_{i+1,j}}{2I_{i-1,j} + 2I_{i+1,j} - 4I_{i,j}} \quad (2.4)$$

$$\Delta Y = \frac{I_{i,j-1} - I_{i,j+1}}{2I_{i,j-1} + 2I_{i,j+1} - 4I_{i,j}} \quad (2.5)$$

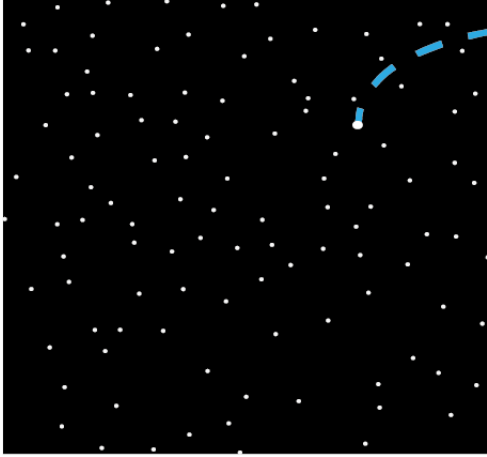
In this way, a matrix of particles and their  $x$  and  $y$  positions is obtained for each image. For more detail, the reader can check the function *CountParticles.m* in the Appendix, which performs the task just described.

### 2.1.3 Particle Matching

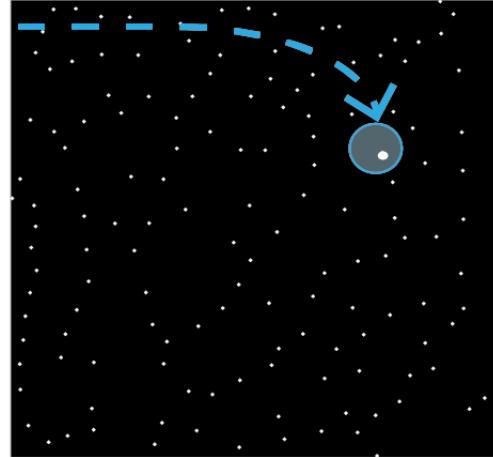
This step is critical since it involves matching the particles found on both frames for each image pair. The search process is implemented efficiently using a k-d tree algorithm found on the web ([MATLAB k-d tree](#)) implemented in the function *ReadCoord.m* in the Appendix. The process followed to get successful matches can be seen in Figure 2.4. For each particle location in the first exposure, the algorithm looks for a particle on the second exposure around that location. More specifically, it uses a search radius which is an input to the function and has to be of the same order of the actual displacement for the process to be successful. In this way, if just one particle is found within the search area in the second exposure, that is the right match. This need to identify all the particles and all the pairs on a one-by-one basis is the main reason why PTV requires a lower image density than PIV.

At this point, it is worth mentioning that although this search algorithm is not really sophisticated, more advanced ones can be easily implemented by using the super-resolution approach proposed by [Keane et al. \(1995\)](#), in which a predictor is built using standard cross-correlation and particle matches can then be sought by restricting the search area.

**Frame A**



**Frame B**

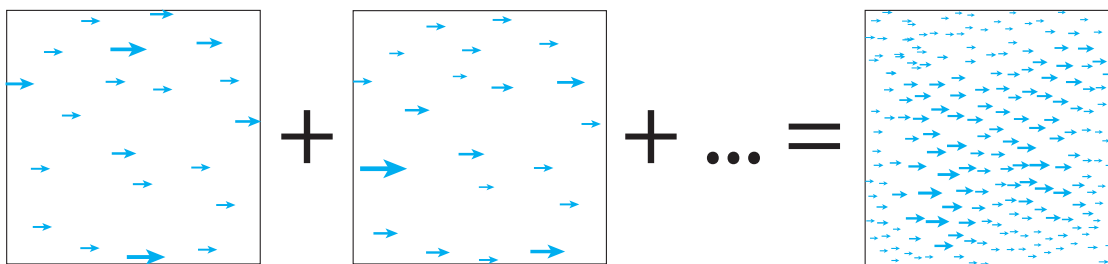


**Figure 2.4:** Sketch of the particle pairing process.

Once the particle pairs have been identified, a displacement vector can be drawn from the particle's location on the first image of the pair to the location on the second image of the pair. Knowing the time interval between the exposures, the displacement vectors can be turned into velocity vectors.

### 2.1.4 Ensemble Creation

For each image pair, the number of independent velocity vectors and hence the resolution, is determined by the number of particles, which generally depends on the image density allowed. However, with the method presented, increasing the number of vectors and therefore spatial resolution reduces to just adding more image pairs and constructing an ensemble of vectors including all of them, as shown in Figure 2.5.



**Figure 2.5:** Ensemble creation: the velocity vectors from every image pair are all put together in a conglomerate.

### 2.1.5 Ensemble Averaging

Once the information from all images has been compiled into a single ensemble, the next step is to average the velocity field over subzones of this ensemble in order to assign a single vector to each of them. In order to preserve accuracy while

increasing spatial resolution, the area over which the average is performed must contain a large enough number of vectors. That number is just limited by the number of images, so they could be as small as a single pixel or even smaller than that, provided the sufficient number of snapshots is taken. Equation 2.6 shows how to compute the number of snapshots  $S$  required to obtain a number of particle pairs  $P$  in a subarea of dimensions  $A_{sub}$  for a particular value of the seeding density measured in particles per pixel,  $N_{ppp}$ .

$$S = \frac{P}{N_{ppp} A_{sub}} \quad (2.6)$$

For instance, in order to obtain measurements with an uncertainty on the mean,  $\sigma_{\bar{u}} = 0.1 \text{ px}$ , the number of particle pairs needed can be found using Equation 2.7. For a reasonable value for the turbulent fluctuations  $u' \approx 1 \text{ px}$  and for the random uncertainty  $\sigma_{\epsilon} \approx 0.1 \text{ px}$ , the number of particle pairs needed is  $P = 101$ .

$$\sigma_{\bar{u}} = \sqrt{\frac{u'^2 + \sigma_{\epsilon}^2}{P}} \quad (2.7)$$

Then, from Equation 2.6, for a particle density of  $N_{ppp} = 0.01 \text{ ppp}$  and a subarea of a single pixel,  $A_p = 1 \text{ px}$ , an estimation of the number of snapshots is obtained,  $S = 10100$ . As it can be seen,  $S \propto \frac{1}{A_{sub}}$  so it might occur that further reductions on the size of the averaging area are difficult to achieve just because the number of snapshots needed increases rapidly with decreasing area. In 3D, as it will be discussed in detail later on, the situation is worse, since the existence of a third dimension decreases the image density in the volume, being much smaller than the one achieved in planar PIV.

The averaging process then basically consists on analyzing the velocity distribution of the particles inside the area considered and computing its mean and variance, that will correspond respectively to the mean velocity and the variance of the velocity fluctuations, which is related to the Reynolds stresses and the turbulent kinetic energy.

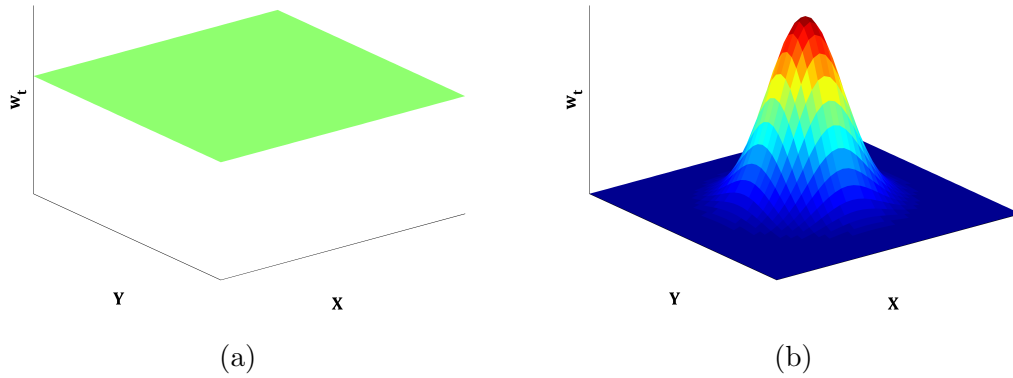
### 2.1.6 Design alternatives: Use of filters

It was observed that the ensemble averaging process tended to smooth sharp velocity gradients in the mean flow field, specially as the averaging window size was increased. This came from the fact that a constant value of the mean flow field was assumed within each averaging window while the real velocity distribution was not necessarily flat. This issue obviously reduced the quality of the mean flow field measurements but also led to the appearance of a residual error that greatly affected the turbulent fluctuations. Although this phenomenon will be thoroughly explained

later on, this brief discussion tries to evidence the existence of the problem and the need for a solution. This work has attempted to tackle this issue and it presents and compares different ideas to optimize the ensemble averaging process in a way that the velocity vector assigned to the area considered best represents the real velocity field at that precise spot. More specifically, the use of filters has been explored, particularly focusing on three of them: top-hat, Gaussian and polynomial. The first two of them rely on the use of weighting windows, basically consisting on performing a weighted average so that each particle does not necessarily contribute by the same amount to the final velocity vector of the region. On the other hand, the polynomial filter tries to fit the velocity distribution into a second order polynomial in order to overcome the deficiencies of the other two methods. The rest of this section will be devoted to the detailed description of the aforementioned filters and *EnsemblePT.m* in the Appendix contains the implementation algorithm for them.

- **Top-hat filter:** This filter corresponds to the regular averaging method in the sense that the weight assigned to each of the particles in the region considered is the same and equal to 1, as described in Equation 2.8 and Figure 2.6a. Therefore, each particle in the subzone equally contributes to the velocity vector assigned to the grid point analyzed.

$$w_{t,i} = 1 \quad (2.8)$$



**Figure 2.6:** Weighting distributions: **a)** Top-hat filter, **b)** Gaussian filter.

- **Gaussian filter:** This filter is based on the notion that the weight of the particles should be somehow related to their distance to the center of the area over which the average is made, which is precisely the grid point where the final averaged velocity vector will be located. More precisely, the weight distribution corresponds to a Gaussian centered at the grid point considered in each case and it is described by Equation 2.9, where  $d_i$  is the distance from particle  $i$  to the grid point  $(X,Y)$ ,  $IV$  is the radius of the subzone and  $\sigma$  is the standard deviation of the Gaussian distribution. For the tests presented in this work, the value of the standard deviation used is  $\sigma = \frac{1}{4}$ . Figure 2.6b offers a visual representation of the distribution.

$$\begin{aligned}
 w_{t,i} &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\left(\frac{d_i}{IV}\right)^2}{2\sigma^2}\right) \\
 d_i &= \sqrt{(x_i - X)^2 + (y_i - Y)^2}
 \end{aligned} \tag{2.9}$$

For the two filters presented up to now, once the weights for every particle are determined, the variables of interest, it is, velocity components and Reynolds stresses, are computed using Equations 2.10 to 2.14.

$$\langle u \rangle = \frac{\sum_{i=1}^N w_{t,i} u_i}{\sum_{i=1}^N w_{t,i}} \tag{2.10}$$

$$\langle v \rangle = \frac{\sum_{i=1}^N w_{t,i} v_i}{\sum_{i=1}^N w_{t,i}} \tag{2.11}$$

$$\langle u'^2 \rangle = \frac{\sum_{i=1}^N w_{t,i}^2 u_i'^2}{\sum_{i=1}^N w_{t,i}^2} = \frac{\sum_{i=1}^N w_{t,i}^2 (u_i - \bar{u})^2}{\sum_{i=1}^N w_{t,i}^2} = \frac{\sum_{i=1}^N w_{t,i}^2 u_i^2}{\sum_{i=1}^N w_{t,i}^2} + \bar{u}^2 - 2\bar{u} \frac{\sum_{i=1}^N w_{t,i}^2 u_i}{\sum_{i=1}^N w_{t,i}^2} \tag{2.12}$$

$$\langle v'^2 \rangle = \frac{\sum_{i=1}^N w_{t,i}^2 v_i'^2}{\sum_{i=1}^N w_{t,i}^2} = \frac{\sum_{i=1}^N w_{t,i}^2 (v_i - \bar{v})^2}{\sum_{i=1}^N w_{t,i}^2} = \frac{\sum_{i=1}^N w_{t,i}^2 v_i^2}{\sum_{i=1}^N w_{t,i}^2} + \bar{v}^2 - 2\bar{v} \frac{\sum_{i=1}^N w_{t,i}^2 v_i}{\sum_{i=1}^N w_{t,i}^2} \tag{2.13}$$

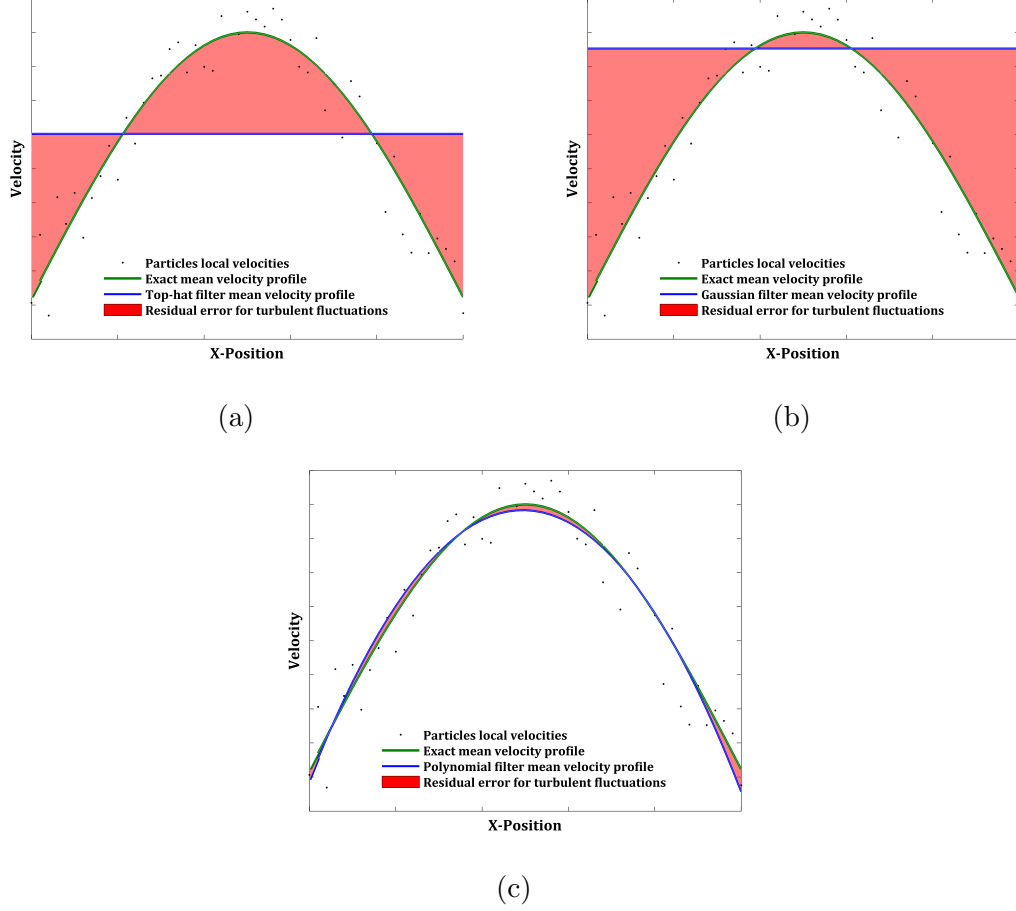
$$\langle u'v' \rangle = \frac{\sum_{i=1}^N w_{t,i}^2 u_i' v_i'}{\sum_{i=1}^N w_{t,i}^2} = \frac{\sum_{i=1}^N w_{t,i}^2 u_i v_i}{\sum_{i=1}^N w_{t,i}^2} + \bar{u}\bar{v} - \bar{v} \frac{\sum_{i=1}^N w_{t,i}^2 u_i}{\sum_{i=1}^N w_{t,i}^2} - \bar{u} \frac{\sum_{i=1}^N w_{t,i}^2 v_i}{\sum_{i=1}^N w_{t,i}^2} \tag{2.14}$$

As previously discussed, the main drawback of the top-hat filter and the Gaussian filter relies on the fact that they do not take into account the changes in the mean flow field within the interrogation window when computing the turbulent fluctuations. Instead, they assume a constant value for the mean velocity all over the window and this has a negative effect on the results. First, it creates a 'smoothing effect' on the mean flow measurements that perturbs the description of

sharp gradients. Secondly, the difference between the mean flow constant value and the actual profile constitutes a residual error for the measurement of the turbulent fluctuations, greatly affecting the quality of the results. Figures 2.7a and 2.7b try to graphically explain this issue in a simplified way, taking into account just one dimension, so that particles with different velocities are scattered along the X-axis. In this way, the particles' location on the sketch is determined by their X-location and its measured speed. In this '1D interrogation window', the final goal is to assign a velocity vector to the point in the center of the window in a way that it best describes the field at that location. In the figures, the mean flow field is depicted with a green line, and not all particles lie on it because some turbulent fluctuations are present. In the case of the top-hat filter, the mean flow velocity of this simplified interrogation window is just the average of the speed of each particle contained on it and it corresponds to the blue line on the sketch. As it can be seen, it underestimates the real speed at that spot. Besides, as a result, when computing the turbulent fluctuations, all the area shaded in red enters in the results as a residual error, since the fluctuations are computed by subtracting a constant value -the blue line in the sketch- to the particles' speed at each location instead of the real mean value of speed at that location -the green line in the sketch-.

For the Gaussian filter depicted in Figure 2.7b, the essence of the problem is exactly the same; however, the effect is not so severe because the Gaussian filter assigns more weight to the particles that are closer to the center of the window. In this way, the smoothing effect on the mean velocity is not that hard and while the residual error increases with the distance from the center of the window, the weight of the particles is inversely proportional and decreases with the distance from the center, so the areas with the largest residual error will have the smallest weights and will have less influence on the final results. However, even though its effects have been mitigated, the problem has not been solved yet and that leads to the introduction of a third filter.

- Polynomial filter:** This filter is conceptually different from the ones presented above. In this case, instead of assigning a certain weight to each particle, it approximates the velocity distribution inside the interrogation window to a second order polynomial. Then, the value of the polynomial fit at the center of the window determines the mean velocity vector associated to that interrogation spot. This avoids the undesired 'smoothing effect' observed when applying the aforementioned filters and allows for a better description of larger gradients. Obviously, a more accurate estimation of the mean flow will generally lead to a better estimation of the Reynolds stresses. Moreover, this polynomial filter seems to have great potential when it comes to measuring turbulence since unlike the top-hat and gaussian filters, it does not assume a constant value for the mean velocity all over the window but instead, it uses the second order polynomial fit to compute the Reynolds stresses, therefore greatly reducing the residual errors mentioned above. An analogous sketch to the one presented for the top-hat and Gaussian filters is shown in Figure 2.7c, denoting the huge reduction on the residual errors.



**Figure 2.7:** Schematic of the residual error formation: **a)** Top-hat filter, **b)** Gaussian filter, **c)** Polynomial filter.

The way to implement this filter mathematically is presented next. For each particle  $N$  in the subzone, its associated speed  $u_N$  is expressed as a second order polynomial (Equation 2.17) which is a function of its distance  $(\Delta x_N, \Delta y_N)$  to the grid point considered,  $(X(i, j), Y(i, j))$ :

$$\Delta x_N = x_N - X(i, j) \quad (2.15)$$

$$\Delta y_N = y_N - Y(i, j) \quad (2.16)$$

$$u_N = a_0 + a_1 \Delta x_N + a_2 \Delta y_N + a_3 \Delta x_N^2 + a_4 \Delta x_N \Delta y_N + a_5 \Delta y_N^2 \quad (2.17)$$

When doing this for every particle in the region, a system of equations is obtained (Equations 2.18 or in a more concise way, Equation 2.19). The



polynomial coefficients  $a_0, a_1, a_2, a_3, a_4$  and  $a_5$  are the unknowns and clearly the system is overdetermined, since the number of particles will exceed the number of unknowns.

$$\begin{bmatrix} u_1 \\ u_2 \\ \dots \\ u_N \end{bmatrix} = \begin{bmatrix} 1 & \Delta x_1 & \Delta y_1 & \Delta x_1^2 & \Delta x_1 \Delta y_1 & \Delta y_1^2 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & \Delta x_N & \Delta y_N & \Delta x_N^2 & \Delta x_N \Delta y_N & \Delta y_N^2 \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} \quad (2.18)$$

$$\bar{u}_{N_px1} = \bar{\bar{M}}_{N_px6} \times \bar{a}_{6x1} \quad (2.19)$$

In order to solve for  $\bar{a}$ , the first step is to premultiply both sides of the equation by the transpose of matrix  $\bar{\bar{M}}$ . In this way, the term that multiplies our unknown  $\bar{a}$ ,  $\bar{\bar{M}}^T \bar{\bar{M}}$  is now a square symmetric matrix and according to the Spectral Theorem, it can be diagonalized and also inverted provided all eigenvalues are real. That is generally the case when enough particles are considered, so it is possible to solve for  $\bar{a}$ , arriving to Equation 2.20.

$$\bar{a} = (\bar{\bar{M}}^T \bar{\bar{M}})^{-1} \bar{\bar{M}}^T \bar{u} \quad (2.20)$$

The same process is followed for all components of velocity; in this 2D case, the steps are repeated to compute a polynomial fit for  $v$ , starting with Equation 2.21 for each particle and arriving at the system in Equation 2.22 that can be solved for  $\bar{b}$ , as shown in Equation 2.23.

$$v_N = b_0 + b_1 \Delta x_N + b_2 \Delta y_N + b_3 \Delta x_N^2 + b_4 \Delta x_N \Delta y_N + b_5 \Delta y_N^2 \quad (2.21)$$

$$\bar{v}_{N_px1} = \bar{\bar{Q}}_{N_px6} \times \bar{b}_{6x1} \quad (2.22)$$

$$\bar{b} = (\bar{\bar{Q}}^T \bar{\bar{Q}})^{-1} \bar{\bar{Q}}^T \bar{v} \quad (2.23)$$

Then, since all coefficients in Equation 2.17 are known and also the equivalent ones for  $v$ , the mean flow components of speed can be obtained just by

equating them to the value of the polynomial at the center of the region, it is, at  $\Delta x = \Delta y = 0$ .

Next, the Reynolds stresses for each particle are obtained using the polynomial fits  $u_{fit}$  and  $v_{fit}$  just computed together with the cloud of measured velocity vectors,  $u_{measured}$  and  $v_{measured}$  in Equations 2.24 to 2.26, where  $N_p$  is the number of particles or equivalently, the number of velocity vectors measured.

$$\langle u'^2 \rangle = \frac{\sum_{i=1}^{N_p} (u_{fit,i} - u_{measured,i})^2}{N_p} \quad (2.24)$$

$$\langle v'^2 \rangle = \frac{\sum_{i=1}^{N_p} (v_{fit,i} - v_{measured,i})^2}{N_p} \quad (2.25)$$

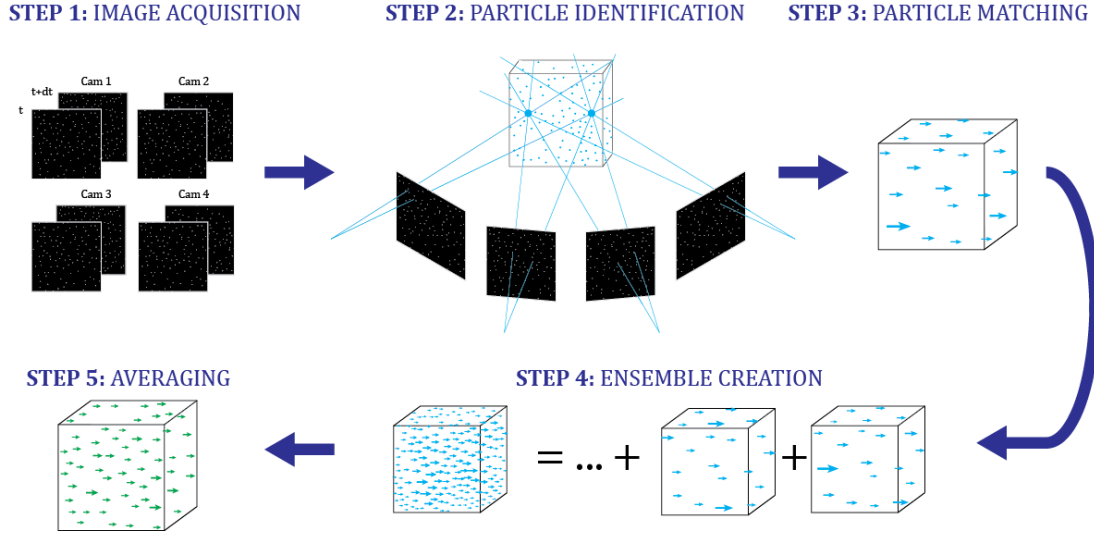
$$\langle u'v' \rangle = \frac{\sum_{i=1}^{N_p} (u_{fit,i} - u_{measured,i})(v_{fit,i} - v_{measured,i})}{N_p} \quad (2.26)$$

## 2.2 3D Ensemble PTV Algorithm

In this section, a three-dimensional extension of the just discussed 2D Ensemble PTV is presented. The process is visually summarized in Figure 2.8 and as it can be seen, most of the steps are just the same from the 2D technique with one more dimension to take into account.

### 2.2.1 Image Acquisition

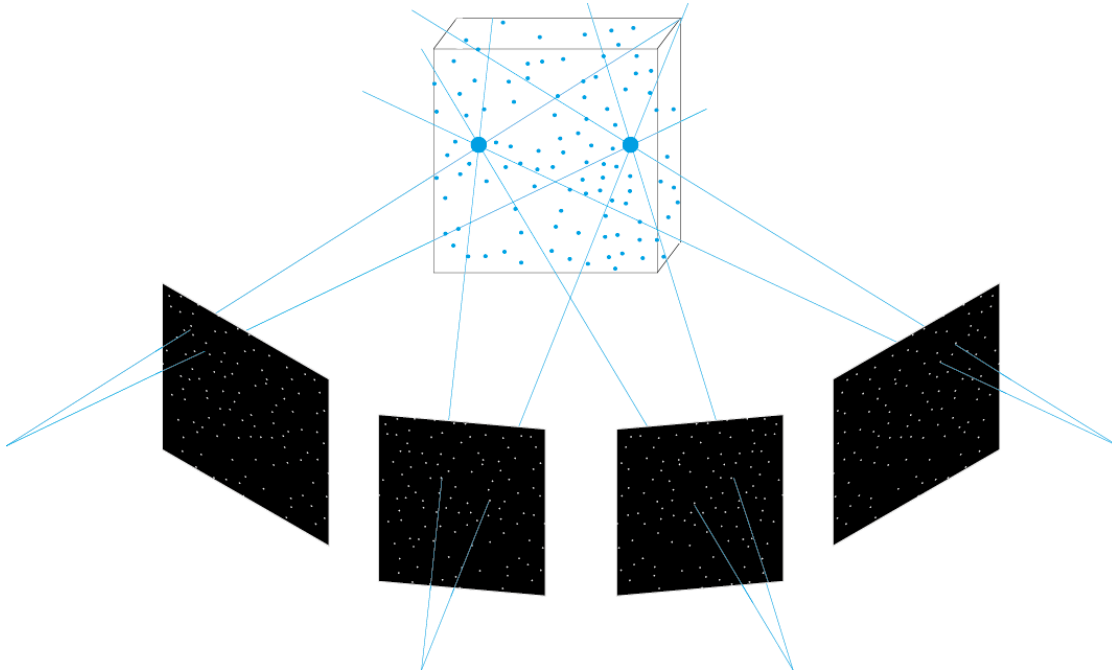
The first step involves obtaining multiple image pairs, each pair consisting on an image at time  $t$  and another one at time  $t+dt$ . In order to get three dimensional results, not just one but several cameras need to record the region of interest from different directions. In this work, since all tests were synthetic, the images were created using the image generation software of the University of Naples which creates the volume seeded with particles and then generates the projections of this volume on each of the cameras.



**Figure 2.8:** Visual representation of the steps involved in the 3D Ensemble PTV method

### 2.2.2 Particle Identification

In 2D, this step reduced to just looking for local maxima on the image intensity map. In 3D, this process is more cumbersome since the 3D coordinates of the particles in the volume of interest must be obtained from 2D images. This is done through triangulation in space: using the particles' projections on each of the images, it is possible to reconstruct the particles' position on the 3D volume, as seen in Figure 2.9. This process has been performed with the triangulation package of the software TPIV (developed at Università di Napoli and UC3M) described in [Discetti and Astarita \(2014\)](#).



**Figure 2.9:** Sketch of the particle identification process.

### 2.2.3 Particle Matching

This step is really similar to its 2D counterpart and it is also performed with the k-d tree algorithm. For each particle located in the 3D volume reconstructed from the first exposure, this algorithm tries to find a particle within a sphere centered at the same point on the second exposure. The size of this sphere is given by the search radius selected and it must be consistent with the particles' displacement for the method to work. If just one particle lies inside the search sphere, then a particle pair is obtained.

Finally, every particle pair defines a displacement vector in 3D space, which can be turned into a velocity vector just by knowing the time interval between the two exposures.

### 2.2.4 Ensemble Creation

At this point, all velocity vectors coming from successive sets of images are integrated into one ensemble. As explained before, this effectively increases the spatial resolution achievable, which basically turns into a function of the number of images processed.

### 2.2.5 Ensemble Averaging

In the last step, velocity vectors are averaged over subvolumes. These subvolumes must contain enough particles so that the resultant velocity vector accurately describes the flow field. Equation 2.27 shows how to compute the number of snapshots  $S$  required to obtain a number of particle pairs  $P$  in a subvolume of dimensions  $V_{sub}$  for a particular value of the seeding density measured in particles per voxel,  $N_{ppv}$ .

$$S = \frac{P}{N_{ppv} V_{sub}} \quad (2.27)$$

Equation 2.27 implicitly assumes that no ghost particles are formed in the process. An estimate of the number of ghost particles is provided by [Discetti and Astarita \(2014\)](#) and has been already discussed in Section 1.2. In order to minimize these spurious matches, the image density should be reduced. A typical figure of merit for the particle density to obtain practically no ghost particles with a 4-camera system is 0.01 *ppv*.

Following the example presented in Section 2.1.5 where 101 particle pairs were needed, and extending it to the three-dimensional case, for a volume depth of  $L_z = 200 \text{ vox}$  and  $N_{ppv} = 0.01 \text{ ppv}$ , then  $N_{ppv} = 0.01/200 \text{ ppv}$  and the number of snapshots needed for a subvolume of dimensions  $V_{sub} = 1 \times 1 \times 1 \text{ vox}$  is  $S = 2020000$ .

As it can be seen, reducing the size of the interrogation windows in 3D is more challenging than in the 2D case. The addition of a third dimension leads to a reduction of the effective particle density and this produces a great increase on the number of snapshots needed. Therefore, it is extremely difficult to work with a subvolume of a single voxel and this evidences the need for the development of flow measurement methods that can provide satisfactory results even when the subvolume size is not that small.

### 2.2.6 Design Alternatives: Use of Filters

The ensemble averaging process in 3D faces the same problems as the 2D counterpart, namely the appearance of a residual error that deteriorates the turbulence statistics. In order to optimize the process, the same three filters used in 2D have been considered again:

- **Top-hat filter:** This filter assigns the same weight to every particle contained within the subvolume considered, as described by Equation 2.28.

$$w_{t,i} = 1 \quad (2.28)$$

- **Gaussian filter:** The implementation of this filter is performed in the same way as in the two dimensional case except for the fact that the Gaussian weighting function is now three-dimensional, as it can be seen in Equations 2.29 and 2.30. Again,  $d_i$  is the distance from particle  $i$  to the grid point  $(X, Y, Z)$ ,  $IV$  is the radius of the subzone and  $\sigma$  is the standard deviation of the Gaussian distribution.

$$w_{t,i} = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\frac{d_i}{IV})^2}{2\sigma^2}\right) \quad (2.29)$$

$$d_i = \sqrt{(x_i - X)^2 + (y_i - Y)^2 + (z_i - Z)^2} \quad (2.30)$$

Once the weights for every particle have been determined, the variables of interest, it is, velocity components and Reynolds stresses, are computed using Equations 2.10 to 2.14 together with Equations 2.31 to 2.34.

$$\langle w \rangle = \frac{\sum_{i=1}^N w_{t,i} w_i}{\sum_{i=1}^N w_{t,i}} \quad (2.31)$$

$$\langle w'^2 \rangle = \frac{\sum_{i=1}^N w_{t,i}^2 w_i'^2}{\sum_{i=1}^N w_{t,i}^2} = \frac{\sum_{i=1}^N w_{t,i}^2 (w_i - \bar{w})^2}{\sum_{i=1}^N w_{t,i}^2} = \frac{\sum_{i=1}^N w_{t,i}^2 w_i^2}{\sum_{i=1}^N w_{t,i}^2} + \bar{w}^2 - 2\bar{w} \frac{\sum_{i=1}^N w_{t,i}^2 w_i}{\sum_{i=1}^N w_{t,i}^2} \quad (2.32)$$

$$\langle u'w' \rangle = \frac{\sum_{i=1}^N w_{t,i}^2 u_i' w_i'}{\sum_{i=1}^N w_{t,i}^2} = \frac{\sum_{i=1}^N w_{t,i}^2 u_i w_i}{\sum_{i=1}^N w_{t,i}^2} + \bar{u}\bar{w} - \bar{w} \frac{\sum_{i=1}^N w_{t,i}^2 u_i}{\sum_{i=1}^N w_{t,i}^2} - \bar{u} \frac{\sum_{i=1}^N w_{t,i}^2 w_i}{\sum_{i=1}^N w_{t,i}^2} \quad (2.33)$$

$$\langle v'w' \rangle = \frac{\sum_{i=1}^N w_{t,i}^2 v_i' w_i'}{\sum_{i=1}^N w_{t,i}^2} = \frac{\sum_{i=1}^N w_{t,i}^2 v_i w_i}{\sum_{i=1}^N w_{t,i}^2} + \bar{v}\bar{w} - \bar{w} \frac{\sum_{i=1}^N w_{t,i}^2 v_i}{\sum_{i=1}^N w_{t,i}^2} - \bar{v} \frac{\sum_{i=1}^N w_{t,i}^2 w_i}{\sum_{i=1}^N w_{t,i}^2} \quad (2.34)$$

- Polynomial filter:** The reasoning behind this filter and the way of implementing it is practically the same as in 2D. The only difference is that the equations used need to be modified to incorporate the third dimension and also some more variables of interest need to be extracted from the analysis. Therefore, for each particle  $N$  in the subzone, its associated speed components  $u_N$ ,  $v_N$  and  $w_N$  are expressed as a second order polynomial (see Equations 2.38 to 2.40) which is a function of its distance  $(\Delta x_N, \Delta y_N, \Delta z_N)$  to the grid point considered,  $(X(i, j, k), Y(i, j, k), Z(i, j, k))$ :

$$\Delta x_N = x_N - X(i, j, k) \quad (2.35)$$

$$\Delta y_N = y_N - Y(i, j, k) \quad (2.36)$$

$$\Delta z_N = z_N - Z(i, j, k) \quad (2.37)$$

$$u_N = a_0 + a_1 \Delta x_N + a_2 \Delta y_N + a_3 \Delta z_N + a_4 \Delta x_N^2 + a_5 \Delta x_N \Delta y_N + a_6 \Delta y_N^2 + a_7 \Delta x_N \Delta z_N + a_8 \Delta y_N \Delta z_N + a_9 \Delta z_N^2 \quad (2.38)$$

$$v_N = b_0 + b_1 \Delta x_N + b_2 \Delta y_N + b_3 \Delta z_N + b_4 \Delta x_N^2 + b_5 \Delta x_N \Delta y_N + b_6 \Delta y_N^2 + b_7 \Delta x_N \Delta z_N + b_8 \Delta y_N \Delta z_N + b_9 \Delta z_N^2 \quad (2.39)$$

$$w_N = c_0 + c_1\Delta x_N + c_2\Delta y_N + c_3\Delta z_N + c_4\Delta x_N^2 + c_5\Delta x_N\Delta y_N + c_6\Delta y_N^2 + c_7\Delta x_N\Delta z_N + c_8\Delta y_N\Delta z_N + c_9\Delta z_N^2 \quad (2.40)$$

A system of  $N_p$  equations with 10 unknowns is obtained for each of the velocity components when the equations for each particle are brought all together. For simplicity, just the  $u$  component will be presented from here; the procedure is identical for the other two components. Expressing the system of equations in matrix form, Equation 2.41 is obtained. The polynomial coefficients  $a_0$  to  $a_9$  are the unknowns and, for a sufficient number of particles, the system is overdetermined and can be solved in the same way it was done in 2D, arriving at Equation 2.42.

$$\bar{u}_{N_px1} = \overline{\overline{M}}_{N_px10} \times \bar{a}_{10x1} \quad (2.41)$$

$$\bar{a} = (\overline{\overline{M}}^T \overline{\overline{M}})^{-1} \overline{\overline{M}}^T \bar{u} \quad (2.42)$$

The same process is followed for all components of velocity to obtain the corresponding polynomial fit. The mean flow components of speed are just the value of each polynomial fit at the center of the subvolume, it is, at  $\Delta x = \Delta y = \Delta z = 0$ . In order to compute the Reynolds stresses for each particle, the polynomial fits  $u_{fit}$ ,  $v_{fit}$  and  $w_{fit}$  just computed and the measured velocity vectors,  $u_{measured}$ ,  $v_{measured}$  and  $w_{measured}$  are used in Equations 2.24 to 2.26 together with 2.43 to 2.45, where  $N_p$  is the number of particles or equivalently, the number of velocity vectors measured.

$$\langle w'^2 \rangle = \frac{\sum_{i=1}^{N_p} (w_{fit,i} - w_{measured,i})^2}{N_p} \quad (2.43)$$

$$\langle u'w' \rangle = \frac{\sum_{i=1}^{N_p} (u_{fit,i} - u_{measured,i})(w_{fit,i} - w_{measured,i})}{N_p} \quad (2.44)$$

$$\langle v'w' \rangle = \frac{\sum_{i=1}^{N_p} (v_{fit,i} - v_{measured,i})(w_{fit,i} - w_{measured,i})}{N_p} \quad (2.45)$$





# Chapter 3

## Results and discussion

The capability and potential of the proposed systems were checked by performing several synthetic tests using virtually-generated images featuring diverse flow fields.

### 3.1 2D Ensemble PTV Algorithm

#### 3.1.1 Validation: Boundary Layer Profile Test

In order to validate the capabilities of the 2D Ensemble PTV method developed, a synthetic test involving a boundary layer flow field was performed. For that, 150 image pairs were created using the traditional equations describing the boundary layer profile on each of the sublayers (Pope, 2000):

- *Viscous sublayer* ( $y^+ < 5$ ): Also called *laminar sublayer*, in this thin region near the wall, viscous effects dominate and the shear stress  $\tau$  can be considered constant. Integrating Equation 3.1 it is possible to arrive at Equation 3.2 that directly expresses the velocity  $u$  as a function of wall units,  $y^+$ , which is a Reynolds number based on the friction velocity  $u_\tau = \sqrt{\frac{\tau_{wall}}{\rho}}$  and the distance from the wall,  $y$ .

$$\tau \cong \tau_{wall} = \mu \frac{\partial u}{\partial y} \Big|_{y=0} \quad (3.1)$$

$$u^+ = \frac{u}{u_\tau} = y^+ \quad (3.2)$$

- *Buffer region* ( $5 < y^+ < 30$ ): The velocity profile in this region was first described by van Driest (1956) making use of the mixing-length hypothesis, arriving at Equation 3.3, where  $l_m$  is the mixing length and the value of  $A^+$

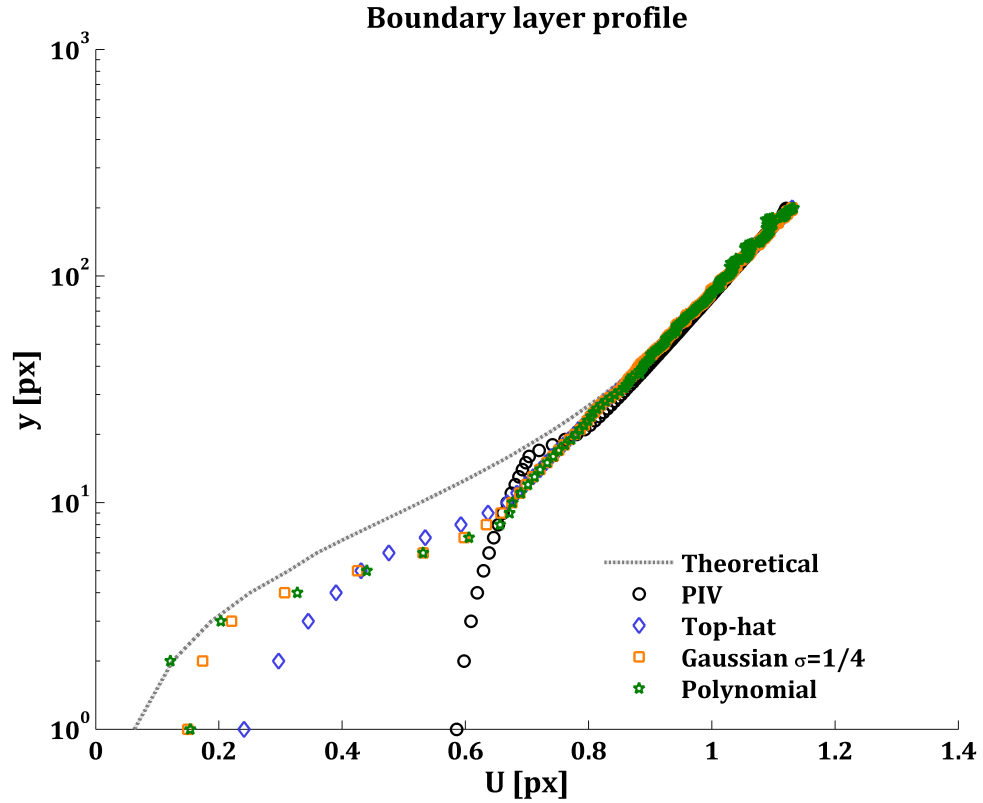
was found experimentally to be  $A^+ = 26$ .

$$u^+ = \int_0^{y^+} \frac{2dy'}{1 + (1 + 4l_m^+(y')^2)^{1/2}} \quad (3.3)$$

$$l_m^+ = \frac{l_m u_\tau}{\nu} = ky^+ \left( 1 - \exp\left(-\frac{y^+}{A^+}\right) \right) \quad (3.4)$$

- *Log law region* ( $y^+ > 30$ ): This layer is dominated by turbulent mixing and it is independent of the external flow. The velocity profile can be described by the *law of the wall*, presented in Equation 3.5 where the values of  $k$  and  $B$  have been found experimentally to be  $k = 0.41$  and  $B = 5.2$ .

$$u^+ = \frac{1}{k} \ln(y^+) + B \quad (3.5)$$



**Figure 3.1:** Validation test results using a Boundary Layer profile flow field.

For the synthetic experiment performed, the size of the wall unit was equated to the pixel size,  $y^+ = 1 \text{ px}$ . The size of the images was  $200 \text{ px} \times 200 \text{ px}$  with a particle density of  $0.01 \text{ ppp}$ . They were processed with the 2D Ensemble PTV software using an interrogation window of radius  $IV = 5 \text{ px}$  and grid distance  $GD = 1 \text{ px}$ , so that approximately 117 particles were found on each window of the ensemble. No noise was included in the images and the particle image diameter was  $2 \text{ px}$ . They were also analyzed using traditional PIV with the same grid distance and a window size of  $32 \text{ px} \times 32 \text{ px}$  in a way that approximately 10 particles are found on each interrogation window for every image pair, which is a good rule of thumb for the correlation to be successful. The results are shown in Figure 3.1. As it can be seen, although all methods replicate the theoretical velocity profile in the log law region, standard PIV is not capable of coping with the large gradients near the wall. On the contrary, 2D Ensemble PTV seems to offer a better solution, especially when used in combination with the Gaussian or polynomial filters. Particularly this last one offers some satisfactory results even inside the viscous sublayer, showing the potential of the method just developed.

### 3.1.2 Performance Assessment of Mean Flow Field Estimation

Once the coherency of the results had been checked, it was time to actually quantify the performance of the method introduced. Since Reynolds stresses results suffer from the residual errors on the mean velocity field, it was decided to first check the accuracy of the mean flow results. For that, 200 image pairs with zero turbulent fluctuations and a sinusoidal mean flow field described by Equation 3.6 were constructed and tested. The size of the images was  $128 \text{ px} \times 128 \text{ px}$  and the particle density was again  $0.01 \text{ ppp}$ . The parameter  $\lambda$  represents the spatial wavelength.

$$u = 0.25 \sin\left(\frac{2\pi y}{\lambda}\right) \quad [\text{px}] \quad (3.6)$$

In order to measure the accuracy of the method, two different magnitudes were computed: the total error  $\delta$  and the Modulation Transfer Fuction (MTF). Knowing the exact flow field at each point  $u_{\text{exact},i}$ , described by Equation 3.6, and the measured field at each point  $u_{\text{measured},i}$ , the total error is a measure of the difference between both, as described in Equation 3.7.

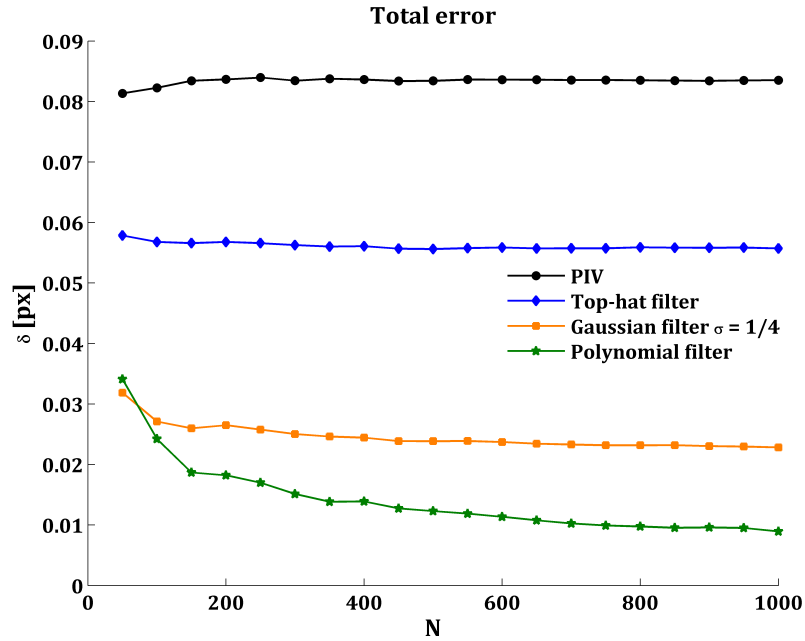
$$\delta = \sqrt{\frac{1}{N} \sum_{i=1}^N (u_{\text{measured},i} - u_{\text{exact},i})^2} \quad (3.7)$$

On the other hand, the MTF is a useful measure of the spatial resolution and it is computed assuming that PIV and PTV only modulate the exact displacement,

so that the measured field still is a sinusoid with the same phase and frequency, but with modulated amplitude. In this way, the total error can be rewritten as in Equation 3.8 by substituting the exact field by its modulated version. Then, the MTF is computed as the value of  $M$  that makes the modulated exact field best match the measured field. This is done by differentiating Equation 3.8 with respect to  $M$  to find the minimum  $\delta^*$ , arriving at Equation 3.9.

$$\delta^* = \sqrt{\frac{1}{N} \sum_{i=1}^N (u_{measured,i} - M u_{exact,i})^2} \quad (3.8)$$

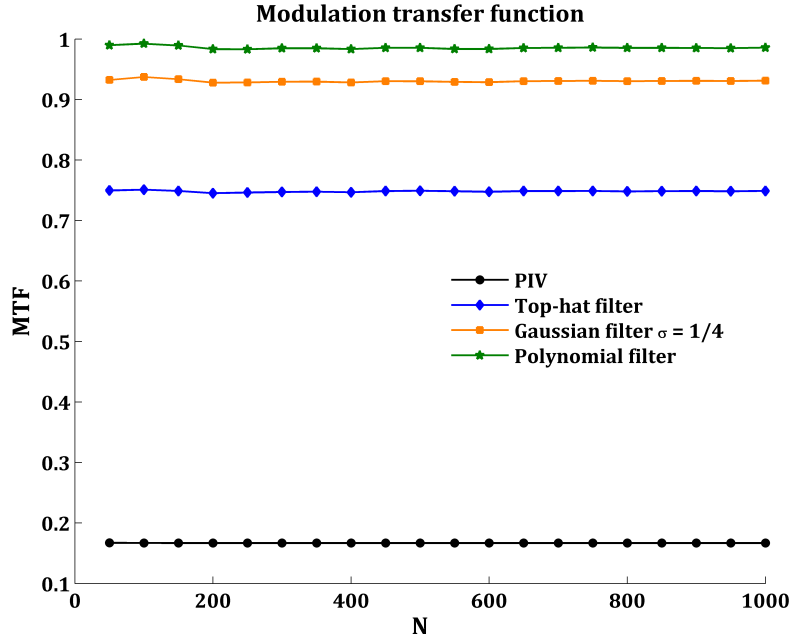
$$M = MTF \leftrightarrow \delta^* = \delta_{min} \Rightarrow MTF = \frac{\sum_{i=1}^N u_{measured,i} u_{exact,i}}{\sum_{i=1}^N u_{exact,i}^2} \quad (3.9)$$



**Figure 3.2:** Total error variation with the number of images processed for a sinusoidal mean flow field.

First of all, these magnitudes were computed for different numbers of images, in order to see how fast each technique converged and how accurate they were. The results obtained are depicted in Figure 3.2 and 3.3 for  $\lambda = 32 \text{ px}$ . The interrogation window size for PIV was  $32 \text{ px} \times 32 \text{ px}$  again, as it is the smallest window that can be used without losing accuracy with the given particle density. On the contrary, for 2D Ensemble PTV, interrogation windows of radius  $IV = 8 \text{ px}$  were used, since this method is able to work with smaller windows without losing accuracy, as explained in the previous section. Therefore, even smaller windows could have

been used just by increasing the number of samples.

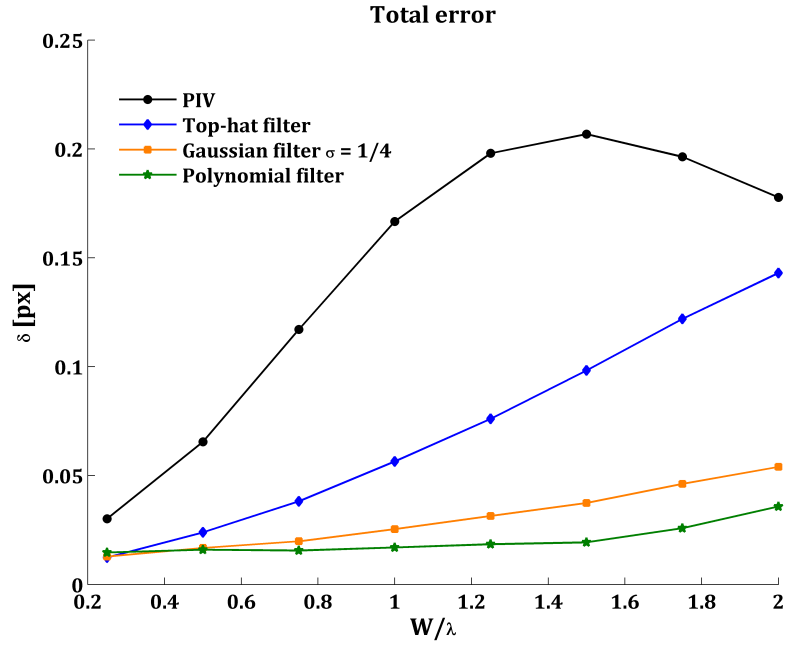


**Figure 3.3:** MTF variation with the number of images processed for a sinusoidal mean flow field.

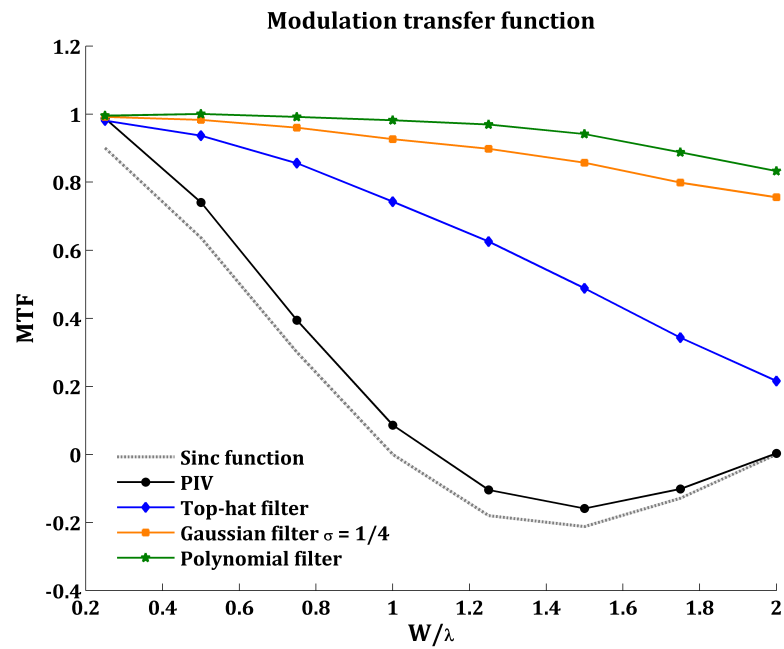
The results are consistent with the performance observed in the previous test. Again, for this value of  $\lambda$ , spatial gradients are quite sharp and Standard PIV cannot negotiate them, providing the largest total error and lowest MTF. Concerning 2D Ensemble PTV, the top-hat filter does not offer really satisfactory results either, although it performs better than PIV. On the contrary, the Gaussian and polynomial filter are really successful. It takes a larger number of images for them to converge to final values because they are more dependent on the number of particles but even with just 50 images they clearly outperform the top-hat filter and PIV, with a MTF of over 90%. Specially the performance of the polynomial filter is quite remarkable, with a total error that ends up being more than two times smaller than the one obtained with the Gaussian filter and with a MTF of almost 100%.

Then, the total error and MTF were computed for different normalised spatial frequencies  $\frac{W}{\lambda}$ , where  $W$  is the size of the interrogation window used for PIV. The size of the images was again  $128 \text{ px} \times 128 \text{ px}$  and the flow field chosen was again the one described by Equation 3.6. The results are shown in Figures 3.4 and 3.5.

The MTF associated with the PIV results should be practically coincident with the one relative to a top-hat moving window filter that, according to [Smith \(1999\)](#), corresponds to  $\text{sinc}(W/\lambda)$ . Therefore, the results presented in Figure 3.5 are consistent, at least for the PIV case, since they match quite well that precise  $\text{sinc}$  function. Aliasing is observed in PIV for  $W/\lambda > 1$  while for the 2D Ensemble PTV algorithm, this phenomenon does not appear, and even at the highest frequencies



**Figure 3.4:** Variation of total error with normalised spatial frequency.



**Figure 3.5:** Variation of MTF with normalised spatial frequency.

the MTF is higher than 80% for the Gaussian and polynomial filters.

Concerning the total error results, they just confirm the superior performance of the polynomial filter and in general of the 2D Ensemble PTV method over traditional PIV for the tests considered.

### 3.1.3 Performance Assessment of Turbulence Estimation.

The last 2D test involved 200 image pairs with sinusoidal mean flow and sinusoidal Reynolds stresses as described in Equations 3.10 and 3.11, where  $\lambda_1 = 32 \text{ px}$  and  $\lambda_2 = 128 \text{ px}$ . A pseudo-turbulence was simulated using the MATLAB function *randn*, which introduces a random fluctuation of known standard deviation on the velocity of the particles. In this case the main focus was to ensure the reliability of the Reynolds stress measurements. The image size, interrogation window size and grid distance were the same as in the previous test presented.

$$\bar{u} = 0.5 \sin\left(\frac{2\pi y}{\lambda_1}\right) \quad [\text{px}] \quad (3.10)$$

$$u' = 0.5 \sin\left(\frac{2\pi y}{\lambda_2}\right) \quad [\text{px}] \quad (3.11)$$

The results are presented in the figures below. Figure 3.6 shows the mean  $u'^2$  profile results together with the exact field. Additionally, Figure 3.7 shows the error maps obtained for the Reynolds stresses using the three different filters presented. It seems that the polynomial filter offers the best solution, estimating the Reynolds stress components with great accuracy even in the presence of sharp gradients. The results obtained using the Gaussian filter are also acceptable, although it overestimates the field at the sine peaks. Finally, the top-hat filter turned out to be too limited to represent this type of flow accurately.

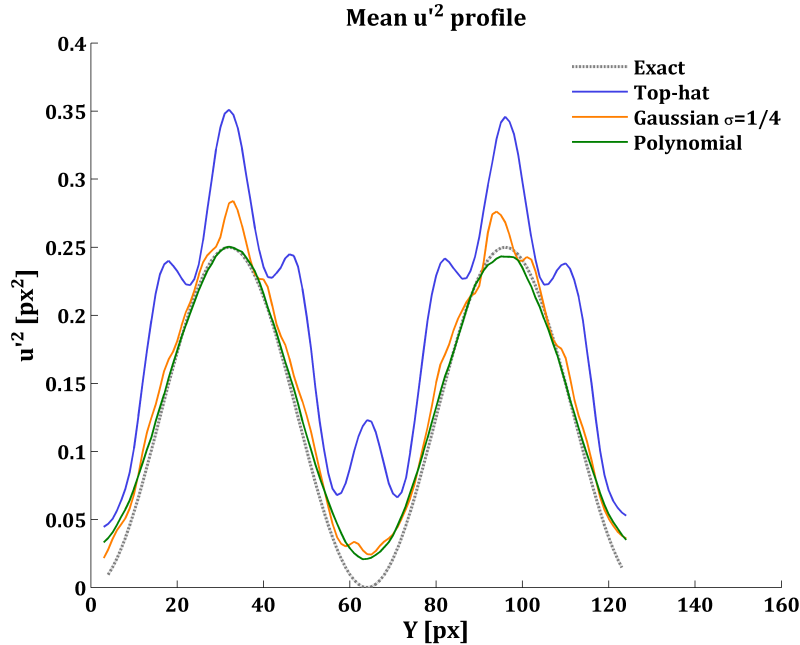
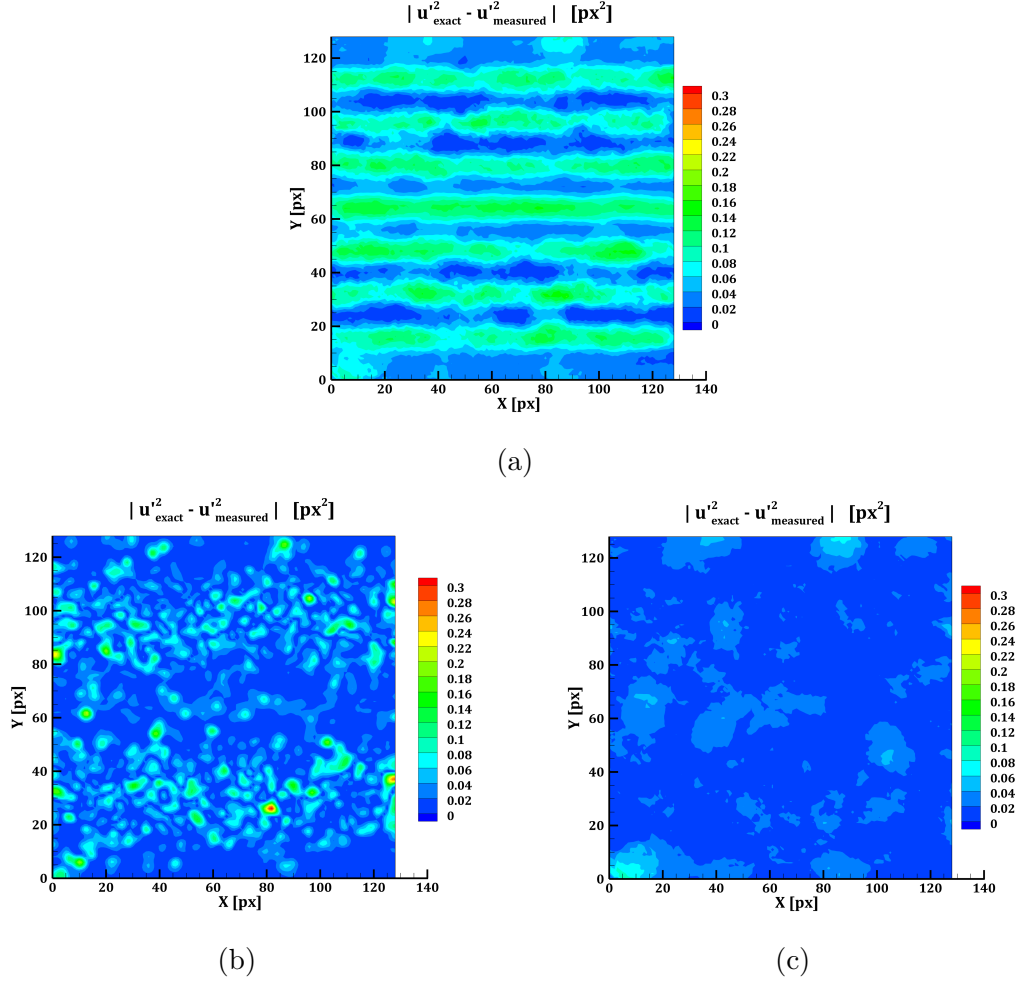


Figure 3.6: Mean  $u'^2$  profile.



**Figure 3.7:** Error maps: **a)** Top-hat filter, **b)** Gaussian filter, **c)** Polynomial filter.

## 3.2 3D Ensemble PTV Algorithm

### 3.2.1 Validation

The 3D Ensemble PTV system's capability was validated by conducting several synthetic tests using four cameras and a  $200 \times 200 \times 200 \text{ vox}$  volume seeded with Gaussian particles. The particle image density was  $0.01 \text{ ppp}$  and a jet-like displacement flow field with a pseudo-shear layer with isotropic turbulence was simulated. The particle distribution was generated with the 3D PIV image generator of the Università di Napoli. Since no instantaneous information was required, the turbulence was generated just by adding noise to the velocity during the particles generation, like it was done in the 2D case. Equations 3.15 to 3.17 describe mathematically the exact flow field, while a visual representation is shown in Figure 3.8. As it can be seen, a round jet was simulated with a maximum displacement of  $3 \text{ vox}$  and a maximum turbulence intensity of  $0.9 \text{ vox}^2$ . The spatial wavelength  $\lambda$  changes from 60 to 90  $\text{vox}$  across the volume studied, as described in Equation 3.12, where  $X_0$  and  $X_{\text{end}}$  correspond to the location of the first and last voxels of the volume. The jet radius  $r_{\text{jet}}$  and the pseudo-shear layer width  $\delta_{\text{SL}}$  are proportional to  $\lambda$  -the jet spreads as  $X$  increases- and are described by Equations 3.13 and 3.14. A polar system of



coordinates centered on the jet axis is used, where  $R = \sqrt{Y^2 + Z^2}$  and  $\theta = \text{atan}(\frac{Y}{Z})$ .

$$\lambda = 60 \left( 1 + 0.5 \frac{X - X_0}{X_{end} - X_0} \right) \quad [vox] \quad (3.12)$$

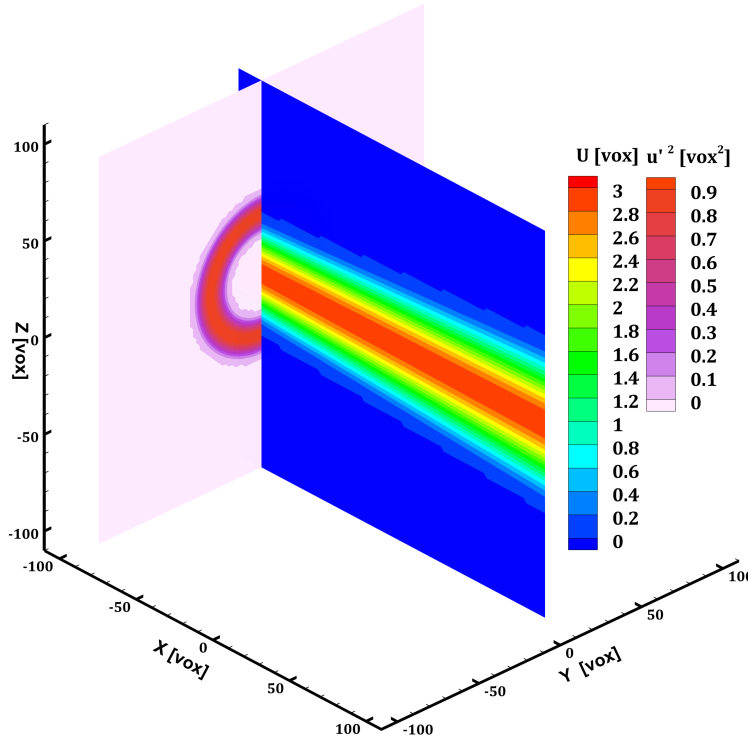
$$r_{jet} = 0.5\lambda \quad [vox] \quad (3.13)$$

$$\delta_{SL} = 0.4\lambda \quad [vox] \quad (3.14)$$

$$U = 1.5 \left( 1 + \cos \left( \frac{2\pi R}{\lambda} \right) \right) \quad [vox] \quad \text{for} \quad R < r_{jet} \quad (3.15)$$

$$V = 0 \quad [vox] \quad (3.16)$$

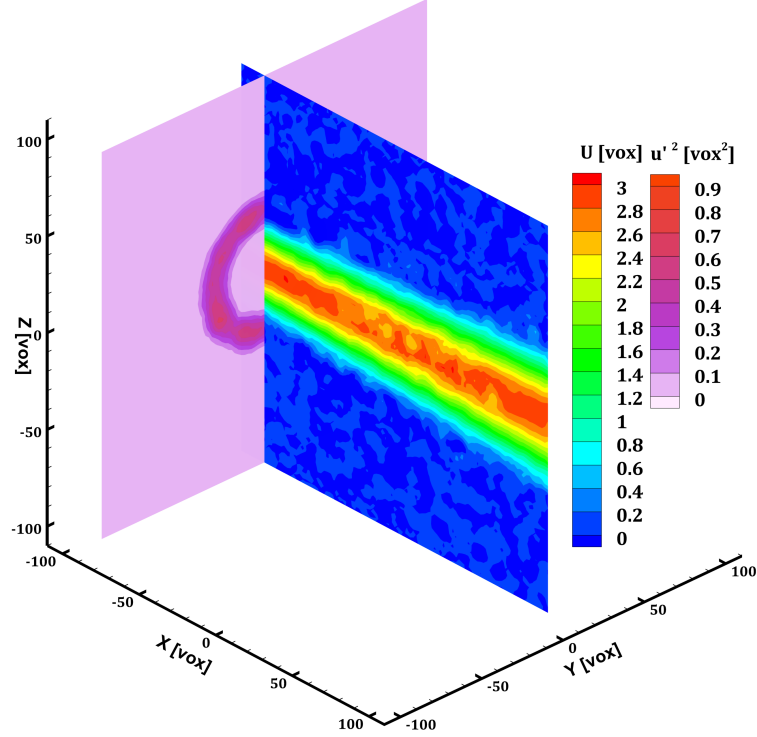
$$u'^2 = v'^2 = w'^2 = 0.45 \left( 1 + \cos \left( \frac{2\pi(R - r_{jet})}{0.4\lambda} \right) \right) \quad [vox^2] \quad \text{for} \quad r_{jet} - \frac{\delta_{SL}}{2} < R < r_{jet} + \frac{\delta_{SL}}{2} \quad (3.17)$$



**Figure 3.8:** Exact jet-like displacement flow field.

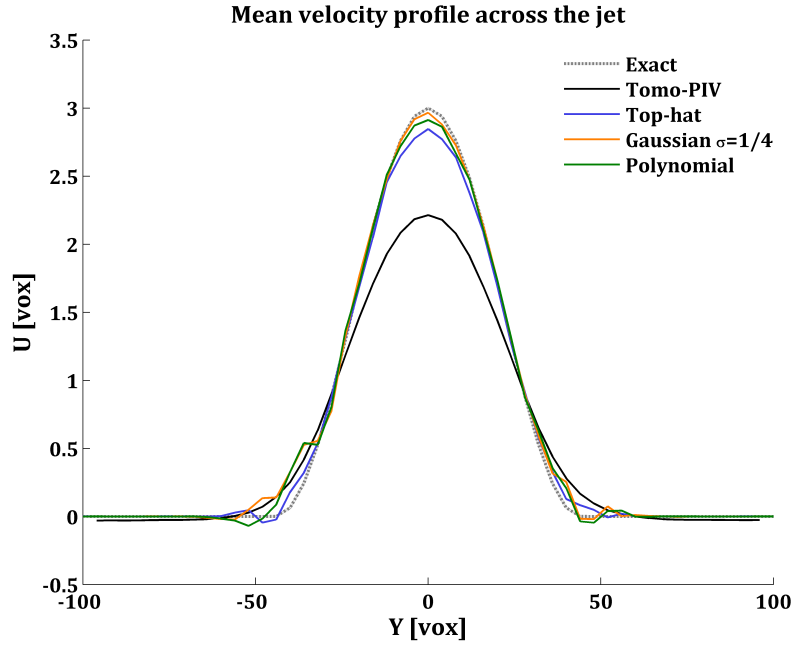
Figure 3.8 depicts the exact flow field used in the simulations, more specifically a slice on the YZ plane showing the Reynolds stress component  $u'^2$  is presented simultaneously with a slice on the XZ plane showing the mean flow field component  $U$ . Figure 3.9 features the same layout but it shows the results obtained after processing 2000 images with the 3D Ensemble PTV technique using the polynomial

filter. The interrogation window had a radius of  $IV = 8 \text{ px}$  and the grid distance was  $GD = 2 \text{ px}$ . Since the image density was  $0.01 \text{ ppp}$ , there were approximately 200 particles on each interrogation subvolume. Just by visual comparison between Figures 3.8 and 3.9 it can be seen that the results are really satisfactory and that the algorithm is able to successfully resolve both the mean flow field and the Reynolds stress components.

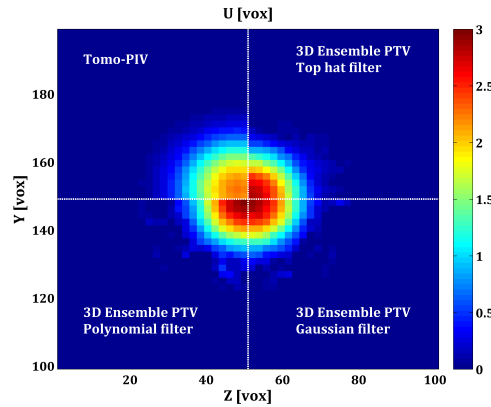


**Figure 3.9:** Results for the jet-like displacement flow field using 3D Ensemble PTV with a polynomial filter.

Figure 3.10 shows the mean flow velocity component  $U$  for the slice on the XY plane containing the jet axis at a given X location. In this case, a comparison between the exact distribution, Tomo-PIV and the three filters of 3D Ensemble PTV is presented. The test settings were the same as the ones mentioned above although Tomo-PIV uses a higher density of  $0.05 \text{ ppp}$ . As mentioned before, in Tomo-PIV, the data is obtained by processing the instantaneous flow fields, and the interrogation window size is related to the particle density in order to assure a successful correlation. In this case, interrogation windows of  $40 \times 40 \times 40 \text{ vox}$  were used for Tomo-PIV, containing 16 particles each, approximately. As it can be seen, Tomo-PIV 'smooths' the velocity gradients in the mean flow field and it is not able to successfully describe the sharp peak in the jet velocity profile. On the contrary, Ensemble PTV offers a much more accurate description. Same conclusions can be drawn from Figure 3.11, that depicts the mean flow field results on a slice on the YZ plane using Tomo-PIV and Ensemble PTV. In this contour, the 'smoothing effect' just discussed can be easily observed in the Tomo-PIV results.



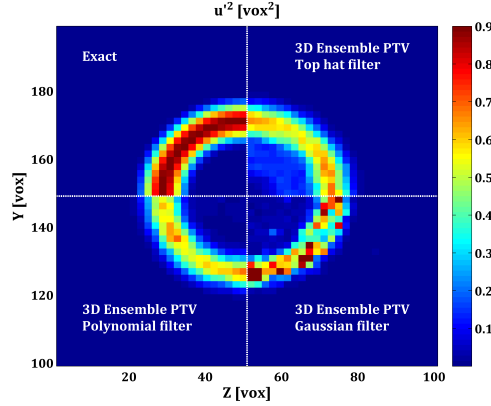
**Figure 3.10:** Mean flow field results for a synthetic test case involving a jet-like displacement on a slice perpendicular to the jet.



**Figure 3.11:** Comparison of results for the mean flow field using Tomo-PIV and Ensemble PTV with three different filters.

Concerning the turbulent fluctuations, Figure 3.12 compares the exact distribution with the results obtained for 3D Ensemble PTV using the three different filters analyzed in this work. The top-hat filter gives acceptable results although it exaggerates the width of the shear layer and the polynomial filter proves to be superior once again. Concerning the Gaussian filter, it is clear that it has some difficulties resolving the turbulent fluctuations and it overestimates them at some points while underestimating them at some others, maybe because the number of particles (hence the number of images) was not enough for this filter to be successful, since it practically neglects the contribution of the particles that are three standard deviations away from the mean. No turbulent fluctuations results are presented using Tomo-PIV because in these synthetic tests, the turbulence

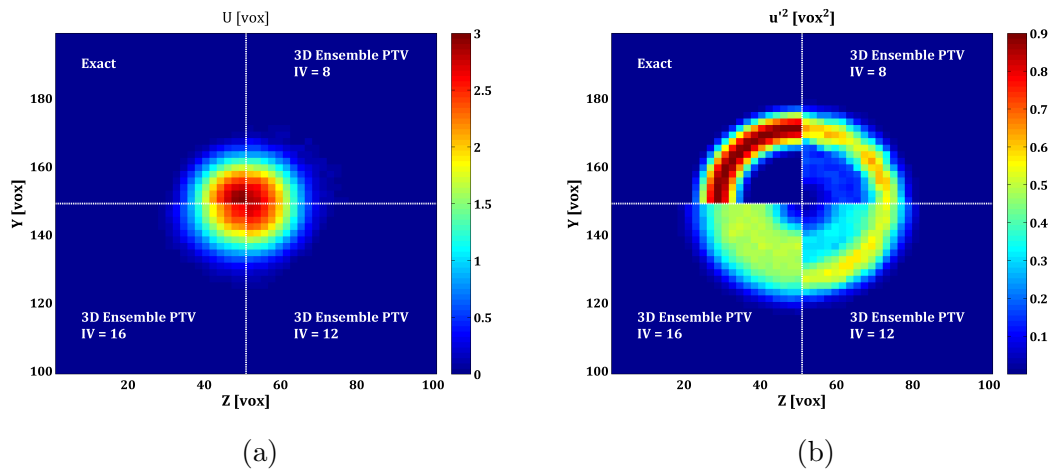
generated is completely random and there is no temporal coherence.



**Figure 3.12:** Comparison of exact distribution of turbulent fluctuations with the results using different filters for the jet-like displacement flow field.

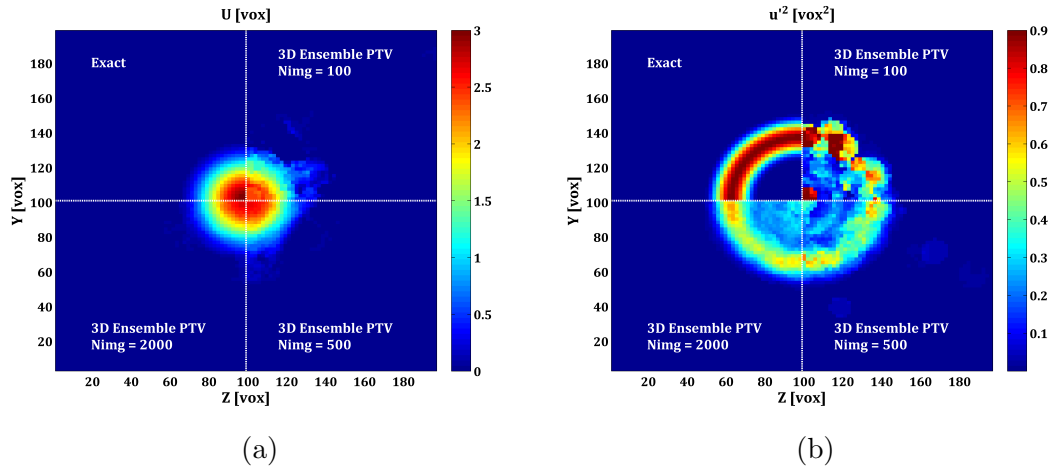
### 3.2.2 Parametric Studies

Finally, the effect of the interrogation window size and the number of images was explored. Figure 3.13 shows the results for the mean flow field and Reynolds stresses when 2000 images are analyzed using 3D Ensemble PTV with a top-hat filter. The radius of the interrogation window was varied from 8 *px* to 16 *px*. This test shows that the proposed technique can provide an accurate estimation of the flow field characteristics when the interrogation window is small enough so that smoothing effects and bias errors are not an issue, but big enough so that a decent amount of particles are contained inside the interrogation volume.



**Figure 3.13:** Results for a synthetic test case involving a jet-like displacement flow field. Influence of the interrogation window size: **a)** Mean flow field, **b)** Reynolds stresses.

Figure 3.14 shows the influence of the number of images on the results when using 3D Ensemble PTV with a top-hat filter and an interrogation window radius of  $8\text{ px}$ . As expected, the accuracy of the results increases rapidly with the number of images. A good representation of the mean flow field can be obtained for a low number of images; however, the turbulent fluctuations suffer from the residual errors on the mean velocity field and as discussed before, they represent the real challenge, so more images are needed to describe them properly.



**Figure 3.14:** Results for a synthetic test case involving a jet-like displacement flow field. Influence of the number of images: **a)** Mean flow field, **b)** Reynolds stresses.



# Chapter 4

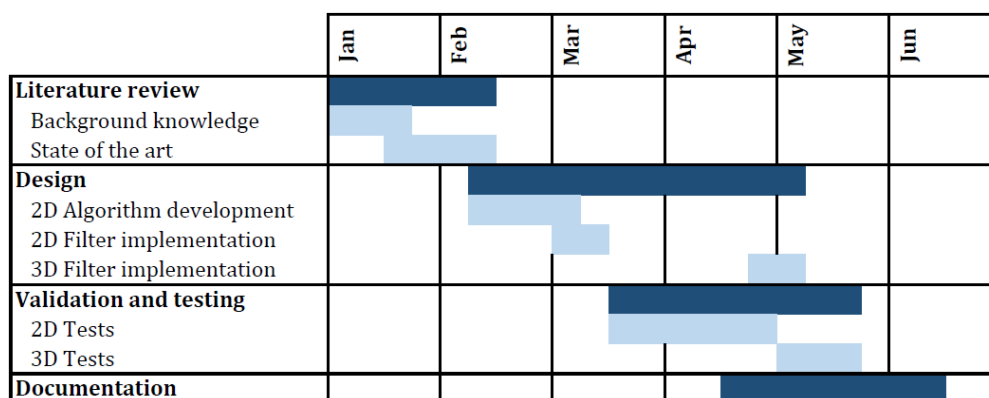
## Project Planning

The development of the project can be divided into four main phases: literature review, design, validation and testing, and documentation. As it can be seen in the Gantt chart presented in Table 4.1, these phases sometimes overlapped during the project time span, which was about six months, with an approximate of 500 hours of work devoted to it. The design process was mostly sequential, fitting into a waterfall-like model.

The literature review phase spanned the first one and a half months approximately and it was devoted to the acquisition of background information necessary to understand the problem and perform a characterization of the state-of-the-art scenario, which was the next stage in the design process.

Once the problem had been identified and understood, the algorithm development and enhancement with filters started and was not completely finished until the beginning of May. At the same time, however, the validation and testing phase was started in parallel in mid-March and it ended in mid-May.

During mid April, once the validity of the algorithm was checked and satisfactory test results were obtained, the documentation phase began and the efforts dedicated to it kept increasing until the end of the project, around mid-June.



**Table 4.1:** Gantt chart for project execution.





# Chapter 5

## Socioeconomic Context and Regulatory Framework

### 5.1 Overview

Turbulence is all around us and it is a well-known fact that its full understanding would lead to the optimization of innumerable processes and designs. Therefore, the development of flow measurement techniques to further investigate the turbulence phenomenon has the potential to bring about a major positive impact in terms of economy, industrial development and environmental protection.

For instance, one of the most important issues concerning the field of aircraft and air transport nowadays is the negative effect this sector has on the environment. According to data provided by Cleansky, a European program devoted to the development of breakthrough technologies to increase the environmental performance in this sector, air transport's contribution to climate change represents 2% of human-induced  $CO_2$  emissions and 12% of all transport sources, while flights produce 628,000,000 tonnes of  $CO_2$  yearly. Turbulence occurs simultaneously with chemical reactions in jet engines used to power aircraft; additionally, the flow over wings and all around the aircraft is turbulent. Therefore, a better understanding of turbulence would lead to more efficient designs, reducing pollution and emissions. Besides, this would decrease the number of design iterations needed and shorten the time to market, leading to more efficient, cleaner and cheaper aircraft. Moreover, it would also help meet the targets set back in 2000 by the Advisory Council for Aeronautical Research in Europe (ACARE) for 2020, such as reducing fuel consumption and  $CO_2$  emissions by 50% per passenger kilometer and reducing  $NO_x$  emissions by 80% . It is clear that providing new instruments to increase our turbulence understanding is mandatory to achieve these goals.

Concerning the regulatory framework, no specific rules apply for this project since all the tests performed were synthetic. However, if the algorithms developed were to be applied to a real experiment, the image acquisition process as well as all the setup preparation would involve the use of lasers that can be extremely hazardous if used improperly. Safety must always be the highest priority and that is why only authorised users who have received the appropriate training on

laser safety can work with these devices. Besides, some safety procedures and rules should be followed, such as wearing appropriate goggles at all times. In the European Community, EN 207 is the European norm for laser safety eyewear.

## 5.2 Project Cost

This section describes all costs associated to the project and proposes an estimate of the budget needed to replicate it. First of all, costs will be divided into direct and indirect costs. Direct costs are the ones that can be specifically attributed to the project developed. Indirect costs include light, Internet connection...and are not that easy to quantify or allocate to the project so they are going to be estimated as 5% of the direct costs.

Concerning direct costs, they can be split into equipment, software, and labor costs. Labor costs are shown in Table 5.1 and they reduce to the salary of an Aerospace Technical Engineer, which has been estimated to be 20€/h, for the duration of the project.

Labor cost			
Item	Price per hour [€/h]	Time [h]	Total [€]
Aerospace Technical Engineer	20	500	10000

**Table 5.1:** Labor cost.

Equipment costs include the amortization cost of a laptop and are contained in Table 5.2 assuming straight line depreciation applies. A depreciation period of 3 years was used, which is consistent with the type of asset considered. For simplicity, the computer dedication will equal the amount of labor hours. The cost of the software needed to perform this project is summarized in Table 5.3.

Equipment cost				
Item	Price [€]	Dedication [h]	Depreciation period [h]	Amortization cost [€]
Laptop	600.00	500	26280	11.42

**Table 5.2:** Equipment cost.

Software cost	
Item	Price [€]
MATLAB for Academic Use	500.00
Windows 8	199.99
<b>Total</b>	<b>699.99</b>

**Table 5.3:** Software cost.

Table 5.4 presents a summary of all the costs included in the project. The final cost is 11246.98 €.

Total Cost	
Direct Cost	
Labor	10000 €
Equipment	11.42 €
Software	699.99 €
Indirect Cost	
Light, Internet,...	535.57 €
<b>Total</b>	<b>11246.98 €</b>

**Table 5.4:** Total cost.



# Conclusion

The main objective of this work was the investigation of Ensemble PTV, specially focusing on its suitability to describe turbulence, which still remains one of the most important unsolved problems in physics. State-of-the-art techniques, such as Tomo-PIV, have proved to be limited in terms of spatial resolution, therefore imposing serious constraints on the description of turbulent flows, and hence, on their ultimate understanding. The huge theoretical potential of Ensemble PTV in terms of spatial resolution seemed to suggest that this technique might be the solution for the current issues and this work tries to prove the validity of that assertion. In order to do that, Ensemble PTV was implemented, validated and tested, both in 2D and 3D. Besides, a possible optimization of the algorithm through the use of filters was examined.

Several synthetic tests were performed: in 2D, a boundary layer profile flow field and a sinusoidal flow field, with and without turbulent fluctuations were tested; in 3D, a jet-like displacement flow field with a pseudo-shear layer with isotropic turbulence was simulated. The results proved the validity of the method proposed and evidenced its superior performance when compared to Standard PIV in 2D and Tomo-PIV in 3D, specially when dealing with sharp velocity gradients. Even more satisfactory results were obtained when combining Ensemble PTV with the use of Gaussian and polynomial filters.

Finally, although the results obtained look really promising and reinforce the idea of Ensemble PTV constituting an actual solution for the current spatial resolution limitations, no further definite conclusions can be made at this point. Real experiments including real noise and all sources of error still need to be performed and only further studies will truly determine if this method can significantly contribute to the achievement of a better description of three-dimensional turbulent flows.



# Bibliography

- Adrian, R. J. (1984), ‘Scattering particle characteristics and their effect on pulsed laser measurements of fluid flow: speckle velocimetry vs particle image velocimetry’, *Appl. Opt.* *23*:1690-1691 .
- Adrian, R. J. (1991), ‘Particle-imaging techniques for experimental fluid mechanics’, *Annu. Rev. Fluid Mech.* *23*:261-304 .
- Arroyo, M. P. and Hinsch, K. D. (2008), ‘Recent developments of PIV towards 3D measurements’, *Topics Appl. Physics* *112*:127-154 .
- Bitter, M., Scharnowski, S., Hain, R. and Kähler, C. J. (2011), ‘High-repetition-rate PIV investigations on a generic rocket model in sub- and supersonic flows.’, *Exp. Fluids* *50*:1019-1030 .
- Brücker, C. (1995), ‘Digital-particle-image-velocimetry (DPIV) in a scanning light-sheet: 3-D starting flow around a short cylinder’, *Exp. Fluids* *19*:255-263 .
- Brücker, C. (1997), ‘3-D scanning PIV applied to an air flow in a motored engine using digital high-speed video’, *Meas. Sci. Technol.* *8*:1480-92 .
- Burgmann, S., Brücker, C. and Schroder, W. (2006), ‘Scanning PIV measurements of a laminar separation bubble’, *Exp. Fluids* *41*:319-26 .
- Collier, R. J., Burckhardt, C. B. and Lin, L. H. (1971), *Optical holography*, Academic Press.
- Discetti, S. (2013), Tomographic Particle Image Velocimetry, PhD thesis, University of Naples Federico II.
- Discetti, S. and Astarita, T. (2014), ‘The detrimental effect of increasing the number of cameras on self-calibration for Tomographic PIV’, *Meas. Sci. Technol.* *25*:084001 .
- Elsinga, G. E., Scarano, F., Wieneke, B. and Van Oudheusden, B. W. (2006), ‘Tomographic particle image velocimetry’, *Exp. Fluids* *41*:933-947 .
- Elsinga, G. E., Westerweel, J., Scarano, F. and Novara, M. (2011), ‘On the velocity of ghost particles and the bias errors in Tomographic-PIV’, *Exp. Fluids* *50*:825-838 .
- Hinsch, K. D. (2002), ‘Holographic particle image velocimetry’, *Meas. Sci. Technol.* *13*:R61-72 .

- Kähler, C. J., Scharnowski, S. and Cierpka, C. (2012), ‘On the resolution limit of digital particle image velocimetry’, *Exp. Fluids* 52:1629-1639 .
- Kähler, C. J. and Scholz, U. (2006), ‘Transonic jet analysis using long-distance micro PIV.’, *12th Int. Symp. on flow visualization (Göttingen, Germany)* .
- Kähler, C. J., Scholz, U. and Ortmanns, J. (2006), ‘Wall-shear-stress and near-wall turbulence measurements up to single pixel resolution by means of long-distance micro-PIV’, *Exp. Fluids* 41:327-341 .
- Keane, R. D., Adrian, R. J. and Zhang, Y. (1995), ‘Super-resolution particle imaging velocimetry’, *Meas. Sci. Technol* 6:754-768 .
- Maas, H. G., Gruen, A. and Papantoniou, D. (1993), ‘Particle tracking velocimetry in three-dimensional flows’, *Exp. Fluids* 15:133-146 .
- Malik, N. A., Dracos, T. and Papantoniou, D. A. (1993), ‘Particle tracking velocimetry in three-dimensional flows’, *Exp. Fluids* 15:279-294 .
- Meng, H., Pan, G., Pu, Y. and Woodward, S. H. (2004), ‘Holographic particle image velocimetry: from film to digital recording’, *Meas. Sci. Technol.* 15:673-685 .
- Olsen, M. G. and Adrian, R. J. (2000), ‘Out-of-focus effects on particle image visibility and correlation in microscopic particle image velocimetry’, *Exp. Fluids* 29:S166-S174 .
- Pope, S. B. (2000), *Turbulent flows*, Cambridge University Press.
- Raffel, M., Willert, C. E., Weerely, S. T. and Kompenhans, J. (2007), *Particle Image Velocimetry: A practical guide*, Springer-Verlag.
- Schaefer, L., Goebbert, J. H., Klaas, M. and Schroeder, W. (2011), ‘Comparison of Holographic and Tomographic Particle-Image Velocimetry turbulent channel flow measurements.’, *9th Int. Symp. PIV (Kobe, Japan)* .
- Scharnowski, S., Hain, R. and Kahler, C. J. (2012), ‘Reynolds stress estimation up to single-pixel resolution using PIV-measurements’, *Exp. Fluids* 52:985-1002 .
- Schnars, U. and Jüptner, W. (1994), ‘Direct recording of holograms by a CCD target and numerical reconstruction’, *Appl. Opt.* 33:179-81 .
- Smith, S. W. (1999), *The scientist and engineer’s guide to digital signal processing*, California Technical Publishing, San Diego.
- Westerweel, J., Geelhoed, P. F. and Lindken, R. (2004), ‘Single-pixel resolution ensemble correlation for micro-PIV applications’, *Exp. Fluids* 37:375-384 .
- Willneff, J. and Gruen, A. (2002), ‘A new spatio-temporal matching algorithm for 3D-particle tracking velocimetry’, *9th International Symposium on Transport Phenomena and Dynamics of Rotating Machinery, Honolulu, Hawaii* .



# Acknowledgements

First of all, I would like to thank Prof. Stefano Discetti, not only for his ideas and fundamental role in making this project become a reality, but also for his inestimable patience, constant encouragement and dedication.

These past four years at University Carlos III have been a truly inspiring experience that granted me the opportunity to meet brilliant faculty, grow personally as well as academically and even spend a year abroad at Purdue University. Not to mention how lucky I feel for all the friends I got to make and with whom I have spent countless hours of class and lab practices and most importantly, of laughter, fun and mutual support. University can be challenging sometimes but it is definitely easier when you are surrounded by such amazing people.

Finally, I would like to have some words for my family and friends back home for their care and support. Special thanks go to my parents, my ultimate role models; and my brother, wherever you are, not a day goes by that I don't think about you and I know you will always look after me. You three have always constituted my world and have made me the person I am today. I owe you everything and no words can express my gratitude for your unconditional guidance, advice and love.



# Appendix

## 2D Ensemble PTV Algorithm - MATLAB Code

### Main script

```
1  clc, clear, close all
2
3  % Filename: Main2D.m
4  % Author: Nereida Aguera
5  % Date: 16/06/2015
6
7  % Main2D.m is the script that executes the 2D Ensemble PTV
   algorithm, calling the appropriate functions for each of
   the 5 steps:
8
9  % STEP 1: Particle Generation (GenImg.m)
10 % STEP 2: Particle Identification (CountParticles.m)
11 % STEP 3: Particle Matching (ReadCoord.m)
12 % STEP 4: Ensemble Creation (EnsemblePT.m)
13 % STEP 5: Ensemble Averaging (EnsemblePT.m)
14
15
16 % PARAMETERS and CONSTANTS
17
18 NImg=1:200;      % Number of images to be analyzed
19 W=128;           % Img Width [px]
20 H=128;           % Img Height [px]
21
22 Res=1;           % Resolution [px/mm]
23 IMax=100;        % Maximum particle intensity
24 DPart=2;         % Particle diameter [px]
25 noise=0;         % Noise
26 DiamDisp=0;      % std of the dispersion of the diameter.
27 IntDisp=0;       % std of the dispersion of the intensity of
   the particle.
28
29 testnum = 75;    % Test number
30 NPart=164;       % Number of particles
31
```

```

32 Root=strcat('test',num2str(testnum));    % Image name root
33
34 IV=8;                % Radius of the interrogation window [px]
35 GD=1;                % Grid distance [px]
36
37 filter = 5;          % Type of filter:
38                        %    0 Top-hat
39                        %    1 - Linear
40                        %    2 - Gaussian sigma = 1/4
41                        %    3 - Gaussian sigma = 1/2
42                        %    5 - Polynomial
43
44 flag = 1;            % Flag for the selection of the flow field:
45                        %    1 - Sinusoidal mean flow field with no
                        %           turbulence
46                        %    2 - Boundary layer profile
47                        %    3 - Sinusoidal mean flow field with
                        %           sinusoidal turbulent fluctuations
48
49 %% STEP 1: Image Generation
50
51 GenImg(Root,W,H,IMax,noise,NPart,DPart,NImg,DiamDisp,IntDisp
52         ,flag)
53
54 %% STEPS 2 and 3: Particle Identification and Particle
55     Matching
56
57 % Preallocation for speed
58 Posa = zeros(NPart,NImg(end));
59 Posb = zeros(NPart,NImg(end));
60 Data = cell(1,NImg(end));
61 n=0;
62
63 addpath('./img')
64 for i = NImg
65     % Read images
66     ImgU8a = imread([Root strcat('_',',', sprintf('%03d',i))
67                     'a.tif'],'tif');
68     ImgU8b = imread([Root strcat('_',',', sprintf('%03d',i))
69                     'b.tif'],'tif');
70
71     % Identify particles on each exposure
72     [~,Posa,~]=CountParticles(ImgU8a,IMax);
73
74     [Npb1,Posb,Ib1]=CountParticles(ImgU8b,IMax);
75
76     % Create particle matches
77     [Matches]=ReadCoord(NImg,Posa,Posb,Res);

```

```
76
77     Data{i}=Matches;
78 end
79
80 % Save particle pairs
81 save(strcat('Data_test',num2str(testnum)),'Data')
82
83
84
85 %% STEPS 4 and 5: Ensemble Creation and Ensemble Averaging
86
87 % Load particle pairs
88 load(strcat('Data_test',num2str(testnum)))
89
90 %Name of the output file
91 NameOut = sprintf('Test%d_%dIV_%dimg_step%d.plt',testnum,2*
92     IV,NImg(end),GD);
93
94 % Ensemble process
95 [Ensemble] = EnsemblePT(NImg(end),NameOut,IV,GD,Data,H,W,
96     filter);
97
98 % Save results
99 save(strcat('Ensemble_test',num2str(testnum)),'Ensemble')
```

## STEP 1: Image Acquisition / Image Generation

```

1  function GenImg(Root,W,H,IMax,noise,NPart,DPart,inum,
2      DiamDisp,IntDisp,flag)
3
4  % Filename: GenImg.m
5  % Author: Nereida Aguera (Adapted from an already existing 3
6      D code).
7  % Date: 16/06/2015
8
9  % This function generates the 2D image pairs.
10
11 % INPUTS:
12 %     Root - String containing the root of the name of the
13 %           image files.
14 %     W - Width of the image [px]. Dimensions [1x1]
15 %     H - Height of the image [px]. Dimensions [1x1]
16 %     IMax - Maximum particle intensity. Dimensions [1x1]
17 %     noise - random noise intensity. Dimensions [1x1]
18 %     DPart - Particles diameter [px]. Dimensions [1x1]
19 %     inum - Sequential number of the output file.
20 %           Dimensions [1x1]
21 %     DiamDisp - std of the dispersion of the diameter.
22 %           Dimensions [1x1]
23 %     IntDisp - std of the dispersion of the intensity of
24 %           the particle. Dimensions [1x1]
25 %     flag - Flag for the selection of the flow field:
26 %           1 - Sinusoidal mean flow field with no
27 %           turbulence
28 %           2 - Boundary layer profile
29 %           3 - Sinusoidal mean flow field with sinusoidal
30 %           turbulent fluctuations
31
32 for i = 1:inum(end)
33     if nargin==0
34         Root='test1_';
35         W=256;
36         H=256;
37         NPart=500;
38         IMax=100;
39         DPart=2;
40         noise=0;
41         inum=i;
42         DiamDisp=0;
43         IntDisp=0;
44     end
45     load('errfun');
46     SQR2=sqrt(2);
47     inum = i;

```

```

41     Img1=zeros(H,W)+randn(H,W)*noise;
42     Img2=zeros(H,W)+randn(H,W)*noise;
43
44     % Particles generation
45     x1=(W-1)*rand(NPart,1)+1;
46     y1=(H-1)*rand(NPart,1)+1;
47     Diam=DPart-DiamDisp+2*rand(NPart,1)*DiamDisp;
48     Int=IMax-IntDisp+2*IntDisp*rand(NPart,1);
49
50     [x1,y1,x2,y2]=VelField(x1,y1,flag);
51
52     DatiPart1=[x1(:) y1(:) Diam(:) Int(:)];
53     DatiPart2=[x2(:) y2(:) Diam(:) Int(:)];
54
55
56     for k=1:NPart
57
58         % Central position of the particle
59         X1=DatiPart1(k,1);
60         X2=DatiPart2(k,1);
61         Y1=DatiPart1(k,2);
62         Y2=DatiPart2(k,2);
63         DP=DatiPart1(k,3);
64         IM=DatiPart1(k,4);
65
66         DpartErf=2*SQR2/(DP);
67
68         IpartInt1=IM;
69         IpartInt2=IM;
70
71         RaggioIntX=floor((3.8*(DP)*0.5)+0.5)+1;
72
73         iMin=floor(Y1+0.5)-RaggioIntX;
74         iMax=floor(Y1+0.5)+RaggioIntX;
75         jMin=floor(X1+0.5)-RaggioIntX;
76         jMax=floor(X1+0.5)+RaggioIntX;
77
78         xabi1=((iMin:iMax)-Y1-.5)*DpartErf;
79         xabi2=((iMin:iMax)-Y1+.5)*DpartErf;
80         dum1=interp1(xerf,ERF,xabi1);
81         dum2=interp1(xerf,ERF,xabi2);
82         dumidiff=dum2-dum1;
83         ivec=iMin:iMax;
84         jvec=jMin:jMax;
85
86         xabj1=((jMin:jMax)-X1-.5)*DpartErf;
87         xabj2=((jMin:jMax)-X1+.5)*DpartErf;
88         dumj1=interp1(xerf,ERF,xabj1);
89         dumj2=interp1(xerf,ERF,xabj2);
90         dumjdiff=dumj2-dumj1;

```

```
91
92
93     for i=1:length(ivec)
94         Dumij=dumidiff(i).*dumjdiff*IpartInt1;
95         for j=1:length(jvec)
96             if(ivec(i)>0 && ivec(i)<=H && jvec(j)>0 &&
97                 jvec(j)<=W)
98                 Img1(ivec(i),jvec(j))=Img1(ivec(i),jvec(
99                     j))+(Dumij(j));
100             end
101         end
102     end
103
104     iMin=floor(Y2+0.5)-RaggioIntX;
105     iMax=floor(Y2+0.5)+RaggioIntX;
106     jMin=floor(X2+0.5)-RaggioIntX;
107     jMax=floor(X2+0.5)+RaggioIntX;
108
109     xabi1=((iMin:iMax)-Y2-.5)*DpartErf;
110     xabi2=((iMin:iMax)-Y2+.5)*DpartErf;
111     dumi1=interp1(xerf,ERF,xabi1);
112     dumi2=interp1(xerf,ERF,xabi2);
113     dumidiff=dumi2-dumi1;
114     ivec=iMin:iMax;
115     jvec=jMin:jMax;
116
117     xabj1=((jMin:jMax)-X2-.5)*DpartErf;
118     xabj2=((jMin:jMax)-X2+.5)*DpartErf;
119     dumj1=interp1(xerf,ERF,xabj1);
120     dumj2=interp1(xerf,ERF,xabj2);
121     dumjdiff=dumj2-dumj1;
122
123     for i=1:length(ivec)
124         Dumij=dumidiff(i).*dumjdiff*IpartInt2;
125         for j=1:length(jvec)
126             if(ivec(i)>0 && ivec(i)<=H && jvec(j)>0 &&
127                 jvec(j)<=W)
128                 Img2(ivec(i),jvec(j))=Img2(ivec(i),jvec(
129                     j))+(Dumij(j));
130             end
131         end
132     end
133
134     end
135
136     s1=sprintf('%s_%03da.tif',Root,inum);
137     Img1=uint8(Img1);
```



```

137     imwrite(Img1,strcat('./img/', s1),'tiff','Compression','
138         none');
139     s1=sprintf('%s_%03db.tif',Root,inum);
140     Img2=uint8(Img2);
141     imwrite(Img2,strcat('./img/', s1),'tiff','Compression','
142         none');
143 end

```

```

1  function [x1,y1,x2,y2]=VelField(x1,y1,flag)
2
3  % Filename: VelField.
4  % Author: Nereida Aguera (Adapted from an already existing 3
5  % Date: 16/06/2015
6
7  % VelField.m is the function that generates the distribution
8  % of particles according to the flow field selected.
9
10 % INPUTS:
11 %     x1 - X position of particle pair. Dimensions: [Nx1],
12 %     where N is the number of particles
13 %     y1 - Y position of particle pair. Dimensions: [Nx1]
14 %     flag - Indicator for the selection of the flow
15 %     field. Dimensions: [1x1]
16
17 % OUTPUTS:
18 %     x1 - X position of particles in the first exposure.
19 %     Dimensions: [Nx1]
20 %     y1 - Y position of particles in the first exposure.
21 %     Dimensions: [Nx1]
22 %     x2 - X position of particles in the second exposure.
23 %     Dimensions: [Nx1]
24 %     y2 - Y position of particles in the second exposure.
25 %     Dimensions: [Nx1]
26
27 switch flag
28
29     case 1 % Sinusoidal mean flow field with no turbulence
30         lambda = 32;
31         dum=(0.5)*sin(2*pi*y1/lambda);
32         x1=x1-dum/2;
33         y1=y1;
34         x2=x1+dum;
35         y2=y1;
36
37     case 2 % Boundary layer profile
38         for i = 1:numel(y1)
39             yplus = y1(i);
40             if yplus<=5

```

```

34         uplus = yplus;
35         dum = uplus/16;
36     elseif 5>yplus>30
37         k = 0.41;
38         Aplus = 26;
39         yprime = 0:0.1:yplus;
40         lmplus = k.*yprime.*(1-exp(-yprime/Aplus));
41         uplus = trapz(yprime,2./(1+sqrt(1+4*lmplus.
            ^2)));
42         dum = uplus/16;
43     else
44         k = 0.41;
45         B = 5.2;
46         uplus = (1/k)*log(yplus)+B;
47         dum = uplus/16;
48     end
49     x1out(i)=x1(i)-dum/2;
50     x2(i)=x1out(i)+dum;
51 end
52 y1=y1;
53 y2=y1;
54 x1=x1out;
55
56 case 3 % Sinusoidal mean flow field with sinusoidal
    turbulent fluctuations
57     lambda=32;
58     lambda2=128;
59     dum = 0.5*sin(2*pi*y1/lambda)+0.5*sin(2*pi*y1/
        lambda2)*randn(1,1);
60     x1=x1-dum/2;
61     y1=y1;
62     x2=x1+dum;
63     y2=y1;
64
65 end

```

## STEP 2: Particle Identification

```

1 function [Np,Pos,I]=CountParticles(ImgU8,IMax)
2
3 % Filename: CountParticles.m
4 % Author: Nereida Aguera (Adapted from an already existing 3
5   D code).
6 % Date: 16/06/2015
7
8 % This function performs the particle identification
9   process.
10
11 % INPUTS:
12 %       ImgU8 - Intensity map of the image. Matrix of
13   dimensions [HxW]
14 %       IMax  - Maximum particle intensity. Dimensions: [1x1
15   ]
16
17 % OUTPUTS:
18 %       Np - Number of particle pairs. Dimensions: [1x1]
19 %       Pos - Matrix of dimensions [Nx2] containing the
20   location of the particles
21 %       I - Intensity at the location of the particles.
22   Vector of dimensions [1xN]
23
24 % Read intensity maps
25 [H W]=size(ImgU8);
26 Img = double(ImgU8);
27
28 th=0.05*IMax;
29 Np=0;
30
31 % Find particles
32 ij=find(Img>th);
33 for ii=1:length(ij)
34     [i,j]=ind2sub([H,W],ij(ii));
35     if (i>1 && i<H && j>1 && j<W)
36         dum=Img(i-1:i+1,j-1:j+1);
37         dum2=dum;
38         dum(2,2)=0;
39         if(Img(i,j)>max(dum(:)))
40             % Gaussian interpolation process
41             dum2=log(dum2+0.01);
42             i0=i+(dum2(1,2)-dum2(3,2))/(2*dum2(1,2)+2*dum2
43               (3,2)-4.*dum2(2,2));
44             j0=j+(dum2(2,1)-dum2(2,3))/(2*dum2(2,1)+2*dum2
45               (2,3)-4.*dum2(2,2));
46             Np=Np+1;
47             Pos(Np,2)=i0;

```

```
41         Pos(Np,1)=j0;  
42         I(Np)=Img(i,j);  
43     end  
44 end  
45 end
```

## STEP 3: Particle Matching

```

1 function [Matches]=ReadCoord(NImg,Posa,Posb,Res)
2
3 % Filename: ReadCoord.m
4 % Author: Nereida Aguera (Adapted from an already existing 3
   %   D code).
5 % Date: 16/06/2015
6
7 % This function performs the particle matching process.
8
9 % INPUTS:
10 %     NImg - Number of images analyzed. Vector of
   %   dimensions [1xN] where N is the number of images
11 %     Posa - Position of particles in the first exposure.
   %   Matrix of dimensions [Nx2]
12 %     Posb - Position of particles in the second exposure.
   %   Matrix of dimensions [Nx2]
13 %     Res - Resolution [px/mm]. Dimensions: [1x1]
14
15 % OUTPUTS:
16 %     Matches: Particle matching data. Matrix of
   %   dimensions [Px4] where P is the number of particle pairs
   %   containing the location and velocity of each pair
17
18
19 % PARAMETERS
20
21 RadX=1/Res;          % Search radius [mm]
22
23 % k-d tree function set performs the matchings
24 addpath('.\kdtree\')
25
26 for ii=1:length(NImg)
27
28     X1=Posa;
29
30     X2=Posb;
31
32     tree=kdtree_build(X2);
33
34     NP=size(X1,1);
35     cont=0;
36     VV=[0 0];
37     XV=VV;
38
39     for i=1:NP
40
41         idxs = kdtree_ball_query( tree, X1(i,:),RadX );
42         if (length(idxs)==1)

```

```
43         cont=cont+1;
44         VV(cont,:)=X2(idxs,:)-X1(i,:);
45         XV(cont,:)=0.5.*(X2(idxs,:)+X1(i,:));
46     end
47
48 end
49
50 kdtree_delete(tree);
51
52 Matches(:,1)=XV(:,1);
53 Matches(:,2)=XV(:,2);
54 Matches(:,3)=VV(:,1);
55 Matches(:,4)=VV(:,2);
56
57 end
```

## STEP 4 and 5: Ensemble Creation and Ensemble Averaging

```

1 function [Mat] = EnsemblePT(NImgend,NameOut,IV,GD,Data,Himg,
   Wimg,filter)
2
3 % Filename: EnsemblePT.m
4 % Author: Nereida Aguera
5 % Date: 16/06/2015
6
7 % This function performs the ensemble creation and ensemble
   averaging processes.
8
9 % INPUTS:
10 %     NImgend - Number of images. Dimensions [1x1]
11 %     NameOut - Name of the output file - String
12 %     IV - Radius of the interrogation window [px].
   Dimensions: [1x1]
13 %     GD - Grid distance [px]. Dimensions [1x1]
14 %     Data - Matrix of dimensions [Px4] where P is the
   number of particle matches, containing the location and
   velocity of the pairs
15 %     Himg - Height of the image [px]. Dimensions [1x1]
16 %     Wimg - Width of the image [px]. Dimensions [1x1]
17 %     filter - Type of filter. Dimensions [1x1]
18 %         0 - Top-hat
19 %         1 - Linear
20 %         2 - Gaussian sigma = 1/4
21 %         3 - Gaussian sigma = 1/2
22 %         5 - Polynomial
23
24 % OUTPUTS:
25 %     Mat - Output 3D matrix containing the number of
   particles, mean flow field and Reynolds stresses results
26
27 addpath('.\kdtree\')
28
29 imgstep = NImgend/1;
30 NImg=0:imgstep:NImgend;
31 NImg(1)=1;
32
33 x1=[0 Wimg];
34 y1=[0 Himg];
35
36 xg=x1(1):GD:x1(2);
37 yg=y1(1):GD:y1(2);
38
39 [X Y]=meshgrid(xg,yg);
40 [H W]=size(X);

```

```

41
42 % Preallocation for speed
43 u=0.*X;
44 v=0.*X;
45 u2=0.*X;
46 v2=0.*X;
47 uv=0.*X;
48 NV=0.*X;
49
50 u_wt_sum=0.*X;
51 v_wt_sum=0.*X;
52 u_2_wt_2_sum=0.*X;
53 v_2_wt_2_sum=0.*X;
54 u_wt_2_sum=0.*X;
55 v_wt_2_sum=0.*X;
56 uv_wt_sum=0.*X;
57 wt_sum=0.*X;
58 wt_2_sum=0.*X;
59 uprime2=0.*X;
60 vprime2=0.*X;
61 uvprime=0.*X;
62
63
64 t1=cputime;
65 for jj=1:numel(NImg)-1
66     index=0;
67     % Ensemble creation
68     for j=NImg(jj):NImg(jj+1)
69         Nv=size(Data{j}(:,1),1); % Number of particle pairs
            found in this image
70         XV(index+1:index+Nv,:)=Data{j}(:,1:2); % Matrix
            containing the U and V components of speed for
            the particle matches in the set of images
            considered
71         V(index+1:index+Nv,:)=Data{j}(:,3:4); % Matrix
            containing the X and Y components of the location
            of the particle matches in the set of images
            considered
72         index=index+Nv;
73     end
74     tree = kdtree_build(XV); % Create pointer to the
        coordinates of particles' matches
75
76     % Ensemble average
77     for i=1:H
78         for j=1:W
79             [idxs]=kdtree_ball_query( tree, [X(i,j) Y(i,j)],
                IV); % idxs is a pointer to the coordinates
                that match particles' positions
80             pos=XV(idxs,:); % Position of the particles

```



```

81         dum=V(idxs,:); % Velocity at position of the
           particles
82         if (isempty(dum)) % If no particles found in the
           search area, skip that grid point
83             continue;
84         elseif filter == 0 || filter == 1 || filter == 2
           || filter == 3 % Assign different weights
           depending on the type of filter
85             if filter == 0 % Top-hat filter
86                 wt = ones(size(pos(:,1)));
87             elseif filter ==1 % Linear filter
88                 d = sqrt((pos(:,1)-X(i,j)).^2+(pos(:,2)-
           Y(i,j)).^2);
89                 wt = 1-1/IV.*d;
90             elseif filter == 2 % Gaussian filter sigma =
           1/4
91                 d = sqrt((pos(:,1)-X(i,j)).^2+(pos(:,2)-
           Y(i,j)).^2);
92                 dnorm = d./IV;
93                 sigma = 1/4;
94                 wt = (1/sqrt(2*pi*sigma^2)).*exp(-dnorm.
           ^2./(2*sigma^2));
95             elseif filter == 3 % Gaussian filter sigma =
           1/2
96                 d = sqrt((pos(:,1)-X(i,j)).^2+(pos(:,2)-
           Y(i,j)).^2);
97                 dnorm = d./IV;
98                 sigma = 1/2;
99                 wt = (1/sqrt(2*pi*sigma^2)).*exp(-dnorm.
           ^2./(2*sigma^2));
100         end
101
102
103         wt = [wt wt]; %[w_i w_i]
104         wt_2 = wt.*wt; %[w_i^2 w_i^2]
105         vel_wt_sum = sum(wt.*dum,1); %[sum(u_i w_i)
           sum(v_i w_i)]
106
107         dum2=(wt.*dum).*(wt.*dum); %[ (u_i w_i)^2 (
           v_i w_i)^2]
108         vel_2_wt_2_sum =sum(dum2,1); %[sum((u_i w_i)
           ^2) sum((v_i w_i)^2))]
109
110
111         dum3=[(wt(:,1).*dum(:,1)).*(wt(:,2).*dum
           (:,2))]; %[ (u_i w_i) (v_i w_i)]
112
113         vel_wt_3_sum =sum(dum3,1); %[sum((u_i w_i) (
           v_i w_i))]
114

```

```
115         dum4 = (wt.*wt.*dum);
116         vel_wt_2_sum = sum(dum4,1); %[sum(u_i w_i^2)
            sum(v_i w_i^2)]
117
118         u_wt_sum(i,j)=u_wt_sum(i,j)+vel_wt_sum(1); %
            [sum(u_i w_i)]
119         v_wt_sum(i,j)=v_wt_sum(i,j)+vel_wt_sum(2); %
            [sum(v_i w_i)]
120
121         u_2_wt_2_sum(i,j)=u_2_wt_2_sum(i,j)+
            vel_2_wt_2_sum(1); %[sum(u_i^2 w_i^2)]
122         v_2_wt_2_sum(i,j)=v_2_wt_2_sum(i,j)+
            vel_2_wt_2_sum(2); %[sum(v_i^2 w_i^2)]
123
124         u_wt_2_sum(i,j)=u_wt_2_sum(i,j)+vel_wt_2_sum
            (1); %[sum(u_i w_i^2)]
125         v_wt_2_sum(i,j)=v_wt_2_sum(i,j)+vel_wt_2_sum
            (2); %[sum(v_i w_i^2)]
126
127         uv_wt_sum(i,j)=uv_wt_sum(i,j)+ vel_wt_3_sum
            (1); %[sum((u_i w_i) (v_i w_i))]
128
129         wt_sum(i,j) = wt_sum(i,j)+sum(wt(:,1),1); %[
            sum(w_i)]
130         wt_2_sum(i,j) = wt_2_sum(i,j)+sum(wt_2(:,1)
            ,1); %[sum(w_i^2)]
131
132         NV(i,j)=NV(i,j)+numel(idxs);
133
134         elseif filter==5 % Polynomial filter
135             [velu,velv,velu2,velv2,veluv]=polyfilter(pos
                ,dum,idxs,X(i,j),Y(i,j));
136             u(i,j)=u(i,j)+velu;
137             v(i,j)=v(i,j)+velv;
138             u2(i,j)=u2(i,j)+velu2.^2;
139             v2(i,j)=v2(i,j)+velv2.^2;
140             uv(i,j)=uv(i,j)+veluv;
141             NV(i,j)=NV(i,j)+numel(idxs);
142         end
143
144     end
145
146     clear idxs
147
148
149 end
150
151 end
152
153 kdtree_delete(tree);
```

```

154
155
156
157 if filter == 0 || filter == 1 || filter == 2 || filter == 3
158     U=(u_wt_sum./wt_sum);
159     up=(u_2_wt_2_sum./wt_2_sum+(u_wt_sum./wt_sum).*(
160         u_wt_sum./wt_sum)-2.*(u_wt_sum./wt_sum).*u_wt_2_sum./
161         wt_2_sum);
162 elseif filter==5
163     U = u./NV;
164     up = uprime2./NV;
165 end
166
167 t2=cputime;
168 fprintf('Elapsed time=%1.1fs\t',t2-t1);
169 fprintf('Remaining=%1.1fs\n',(t2-t1)/jj*(numel(NImg)-jj));
170
171 NVar=8;
172
173 % Create output
174 if filter == 0 || filter == 1 || filter == 2 || filter == 3
175     Mat=zeros(H,W,NVar);
176     Mat(:,:,1)=X;
177     Mat(:,:,2)=Y;
178     Mat(:,:,3)=(u_wt_sum./wt_sum);
179     Mat(:,:,4)=(v_wt_sum./wt_sum);
180     Mat(:,:,5)=(u_2_wt_2_sum./wt_2_sum+(u_wt_sum./wt_sum).*(
181         u_wt_sum./wt_sum)-2.*(u_wt_sum./wt_sum).*u_wt_2_sum./
182         wt_2_sum);
183     Mat(:,:,6)=(v_2_wt_2_sum./wt_2_sum+(v_wt_sum./wt_sum).*(
184         v_wt_sum./wt_sum)-2.*(v_wt_sum./wt_sum).*v_wt_2_sum./
185         wt_2_sum);
186     Mat(:,:,7)=(uv_wt_sum./wt_2_sum+(u_wt_sum./wt_sum).*(
187         v_wt_sum./wt_sum))-(v_wt_sum./wt_sum).*u_wt_2_sum./
188         wt_2_sum-(u_wt_sum./wt_sum).*v_wt_2_sum./wt_2_sum;
189     Mat(:,:,8)=NV;
190 elseif filter == 5
191     Mat=zeros(H,W,NVar);
192     Mat(:,:,1)= X;
193     Mat(:,:,2)= Y;
194     Mat(:,:,3)= u./(numel(NImg)-1);
195     Mat(:,:,4)= v./(numel(NImg)-1);
196     Mat(:,:,5)= u2./(numel(NImg)-1);
197     Mat(:,:,6)= v2./(numel(NImg)-1);
198     Mat(:,:,7)= uv./(numel(NImg)-1);
199     Mat(:,:,8)= NV;
200 end
201
202 TituloZona=sprintf('%s\n',NameOut);TituloZona(find(
203     TituloZona==10))=0;

```

```

195 Titolo=sprintf('b16\n');Titolo(find(Titolo==10))=0;
196 NomeVar=sprintf('X\nY\nU\nV\nup\nvp\nupvp\nNV\n');NomeVar(
    find(NomeVar==10))=0;
197 WritePlt(NameOut,NVar,W,H,Mat,Titolo,NomeVar,TitoloZona);
198 end

```

```

1 function [velu,velv,velu2,velv2,veluv,Np]=polyfilter(pos,dum
    ,idxs,X,Y)
2
3 % Filename: polyfilter.m
4 % Author: Nereida Aguera
5 % Date: 16/06/2015
6
7 % This function implements the polynomial filter in the
    ensemble averaging process.
8
9 % INPUTS:
10 %     pos - Matrix containing the X and Y components of
        the location of the particle pair. Dimensions [Px2],
        where P is the number of particle pairs
11 %     dum - Matrix containing the U and V components of
        speed of the particle pair.
12 %     idxs - Pointer to the coordinates that correspond
        the particle matches location
13 %     X - X-coordinate of the grid point considered.
        Dimensions [1x1]
14 %     Y - Y-coordinate of the grid point considered.
        Dimensions [1x1]
15
16 % OUTPUTS:
17 %     velu - U component of the mean flow field.
        Dimensions [1x1]
18 %     velv - V component of the mean flow field.
        Dimensions [1x1]
19 %     velu2 - u' component of the Reynolds stresses.
        Dimensions [1x1]
20 %     velv2 - v' component of the Reynolds stresses.
        Dimensions [1x1]
21 %     veluv - u'v' component of the Reynolds stresses.
        Dimensions [1x1]
22 %     Np - Number of particle pairs. Dimensions [1x1]
23
24
25 Np=length(idxs);
26 M=zeros(Np,6);
27
28 if(Np<8) %If not enough particles are found around that
        grid location, no velocity is assigned to it
29     velu=0;

```

```

30     velv=0;
31     velu2=0;
32     velv2=0;
33     veluv=0;
34     return;
35 end
36
37
38 dx=pos(:,1)-X;
39 dy=pos(:,2)-Y;
40
41 M=[ones(size(dx)), dx,dy,dx.*dx,dx.*dy,dy.*dy];
42
43 u=dum(:,1);
44 v=dum(:,2);
45
46 Mat=inv(M'*M)*(M');
47
48 % Compute polynomial coefficients
49 a=Mat*u;
50 b=Mat*v;
51
52 % Compute velocity components
53 velu=a(1);
54 velv=b(1);
55 velu2=0;
56 velv2=0;
57 veluv=0;
58
59 % Compute polynomial fittings
60 ufit=(a(1)+a(2)*dx+a(3)*dy+a(4)*dx.^2+a(5).*dx.*dy+a(6).*dy.
    ^2);
61 vfit=(b(1)+b(2)*dx+b(3)*dy+b(4)*dx.^2+b(5).*dx.*dy+b(6).*dy.
    ^2);
62
63 % Compute Reynold stresses
64 for i=1:Np
65     velu2=velu2+(dum(i,1)-ufit(i))^2;
66     velv2=velv2+(dum(i,2)-vfit(i))^2;
67     veluv=veluv+(dum(i,2)-vfit(i))*(dum(i,1)-ufit(i));
68 end
69
70 velu2=sqrt(velu2./(Np-1));
71 velv2=sqrt(velv2./(Np-1));
72 veluv=veluv./Np;

```